SYBASE®

Sybase® Adaptive Server™ Enterprise
System Administration Guide

Adaptive Server™

Document ID: 32500-01-1150

September 1997

Copyright Information

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Table of Contents

About This Book

Audience xxxi
How to Use This Book xxxi
Adaptive Server Enterprise Documents xxxiii
Other Sources of Information xxxv
Conventions Used in This Manual xxxvi
Formatting SQL Statements xxxvi
SQL Syntax Conventions xxxvi
Case xxxvii
Obligatory Options {You Must Choose At Least One} xxxvii
Optional Options [You Don't Have to Choose Any] xxxvii
Ellipsis: Do it Again (and Again)... xxxviii
Expressions xxxviii
If You Need Help xxxix

1. Overview of System Administration

Adaptive Server Database Administration Tasks 1-1
Roles Required for System Administration Tasks 1-2
Using *isql* to Perform System Administration Tasks 1-2
Starting *isql* 1-2
Entering Statements 1-3
Saving and Reusing Statements 1-3
Using Sybase Central for System Administration Tasks 1-3
Sybase Central and Transact-SQL Scripts 1-4
System Tables 1-4
Querying the System Tables 1-5
Keys in System Tables 1-6
Updating System Tables 1-6
System Procedures 1-7
Using System Procedures 1-7
System Procedure Tables 1-8
Creating System Procedures 1-9
System Extended Stored Procedures 1-9
Creating System ESPs 1-10
Logging Error Messages 1-10
Connecting to Adaptive Server 1-11
The Interfaces File 1-11

2. System Databases

Overview of System Databases 2-1
***master* Database** 2-2
Controlling Object Creation in *master* 2-3
Backing Up *master* and Keeping Copies of System Tables 2-4
***model* Database** 2-4
***sybsystemprocs* Database** 2-5
***tempdb* Database** 2-6
Creating Temporary Tables 2-6
***sybsecurity* Database** 2-7
***sybsystemdb* Database** 2-8
***pubs2* and *pubs3* Sample Databases** 2-8
Maintaining the Sample Databases 2-8
pubs2 image Data 2-9
***sybsyntax* Database** 2-9
***dbccdb* Database** 2-9

3. System Administration for Beginners

Using "Test" Servers 3-1
Understanding New Procedures and Features 3-2
Planning Resources 3-2
Achieving Performance Goals 3-3
Installing Sybase Products 3-3
Check Product Compatibility 3-3
Install or Upgrade Adaptive Server 3-3
Install Additional Third-Party Software 3-4
Configure and Test Client Connections 3-4
Allocating Physical Resources 3-4
Dedicated vs. Shared Servers 3-5
Decision Support and OLTP Applications 3-5
Plan Resource Usage in Advance 3-6
Configure the Operating System 3-7
Backup and Recovery 3-7
Keep Up-to-Date Backups of Master 3-8
Keep Offline Copies of System Tables 3-8
Automate Backup Procedures 3-9
Verify Data Consistency Before Backing Up a Database 3-10
Monitor the Log Size 3-10
Ongoing Maintenance and Troubleshooting 3-11
Starting and Stopping Adaptive Server 3-11

Viewing and Pruning the Error Log. 3-11
Keeping Records 3-11
Contact Information 3-12
Configuration Information 3-12
Maintenance Schedules 3-12
System Information 3-13
Disaster Recovery Plan 3-13
Getting More Help 3-14

4. Diagnosing System Problems

How Adaptive Server Uses Error Messages to Respond to System Problems 4-1
Error Messages and Message Numbers 4-2
Variables in Error Message Text 4-3
Adaptive Server Error Logging 4-4
Error Log Format 4-5
Severity Levels 4-6
Levels 10–18. 4-7
Level 10: Status Information. 4-7
Level 11: Specified Database Object Not Found 4-8
Level 12: Wrong Datatype Encountered 4-8
Level 13: User Transaction Syntax Error. 4-8
Level 14: Insufficient Permission to Execute Command 4-8
Level 15: Syntax Error in SQL Statement 4-9
Level 16: Miscellaneous User Error 4-9
Level 17: Insufficient Resources 4-9
Level 18: Non-Fatal Internal Error Detected 4-10
Severity Levels 19–24 4-10
Level 19: Adaptive Server Fatal Error in Resource 4-11
Level 20: Adaptive Server Fatal Error in Current Process 4-11
Level 21: Adaptive Server Fatal Error in Database Processes 4-11
Level 22: Adaptive Server Fatal Error: Table Integrity Suspect. . . . 4-11
Level 23: Fatal Error: Database Integrity Suspect 4-12
Level 24: Hardware Error or System Table Corruption 4-12
Level 26: Rule Error 4-12
Reporting Errors 4-12
Backup Server Error Logging 4-13
Killing Processes 4-14
Using *sp_lock* to Examine Blocking Processes. 4-17
Configuring Adaptive Server to Save SQL Batch Text 4-17
Allocating Memory for Batch Text 4-18

Configuring the Amount of SQL Text Retained in Memory 4-19
Enabling Adaptive Server to Start Saving SQL Text 4-20
SQL Commands Not Represented by Text 4-20
Viewing the Query Plan of a SQL Statement 4-22
Viewing Previous Statements 4-22
Viewing a Nested Procedure 4-23
Shutting Down Servers 4-23
Shutting Down Adaptive Server 4-24
Shutting Down a Backup Server 4-24
Checking for Active Dumps and Loads 4-24
Using *nowait* on a Backup Server 4-25
Learning About Known Problems 4-25

5. Overview of Disk Resource Issues

Device Allocation and Object Placement 5-1
Commands for Managing Disk Resources 5-2
Considerations in Storage Management Decisions 5-4
Recovery 5-4
Keeping Logs on a Separate Device 5-4
Mirroring 5-4
Performance 5-5
Status and Defaults at Installation Time 5-5
System Tables That Manage Storage 5-6
The *sysdevices* Table 5-7
The *sysusages* Table 5-8
The *syssegments* Table 5-8
The *sysindexes* Table 5-9

6. Initializing Database Devices

What Are Database Devices? 6-1
Using the *disk init* Command 6-1
***disk init* Syntax** 6-2
disk init Examples 6-3
Specifying a Logical Device Name with *disk init* 6-3
Specifying a Physical Device Name with *disk init* 6-3
Choosing a Device Number for *disk init* 6-3
Specifying the Device Size with *disk init* 6-4
Optional Parameters for *disk init* 6-5
Getting Information About Devices 6-6

Dropping Devices 6-7
Designating Default Devices 6-7
Choosing Default and Nondefault Devices 6-8
Placing Objects on Database Devices 6-9

7. Mirroring Database Devices

What Is Disk Mirroring? 7-1
Deciding What to Mirror 7-1
Mirroring Using Minimal Physical Disk Space 7-2
Mirroring for Nonstop Recovery 7-3
Conditions That Do Not Disable Mirroring 7-5
Disk Mirroring Commands 7-6
Initializing Mirrors 7-6
Effects on System Tables 7-8
Unmirroring a Device 7-8
Temporarily Deactivating a Device 7-8
Permanently Disabling a Mirror 7-9
Effects on System Tables 7-9
Restarting Mirrors 7-10
waitfor mirrorexit 7-10
Mirroring the Master Device 7-11
Getting Information About Devices and Mirrors 7-11
Disk Mirroring Tutorial 7-11

8. Configuring Memory

Maximizing Adaptive Server Memory 8-1
If Adaptive Server Cannot Start 8-2
How Adaptive Server Uses Memory 8-2
System Procedures for Configuring Memory 8-4
Using *sp_configure* to Set Configuration Parameters 8-5
Using *sp_helpconfig* to Get Help on Configuration Parameters 8-6
Using *sp_monitorconfig* to Find Metadata Cache Usage Statistics 8-7
Major Uses of Adaptive Server Memory 8-8
Adaptive Server Executable Code and Overhead 8-8
Data and Procedure Caches 8-9
How Space Is Split Between Data and Procedure Cache 8-9
Monitoring Cache Space 8-10
User Connections 8-12
Open Databases, Open Indexes, and Open Objects 8-12

Number of Locks 8-13
Database Devices and Disk I/O Structures 8-13
Other Parameters That Use Memory 8-13
Parallel Processing 8-13
Worker Processes 8-14
Partition Groups 8-14
Remote Servers 8-14
Number of Remote Sites 8-15
Other Configuration Parameters for RPCs 8-15
Referential Integrity 8-15
Other Parameters That Affect Memory 8-16

9. Configuring Data Caches

The Data Cache on Adaptive Server 9-1
Cache Configuration Commands 9-3
Information on Data Caches 9-4
Configuring Data Caches 9-6
Explicitly Configuring the Default Cache 9-8
Changing a Cache's Type 9-10
Configuring Cache Replacement Policy 9-10
Dividing a Data Cache into Memory Pools 9-11
Matching Log I/O Size for Log Caches 9-14
Binding Objects to Caches 9-15
Cache Binding Restrictions 9-16
Getting Information About Cache Bindings 9-17
Checking Cache Overhead 9-17
How Overhead Affects Total Cache Space 9-18
Dropping Cache Bindings 9-19
Changing the Wash Area for a Memory Pool 9-20
When the Wash Area Is Too Small 9-22
When the Wash Area Is Too Large 9-22
Changing the Asynchronous Prefetch Limit for a Pool 9-23
Resizing Named Data Caches 9-24
Increasing the Size of a Cache 9-24
Decreasing the Size of a Cache 9-25
Dropping Data Caches 9-26
Changing the Size of Memory Pools 9-27
Moving Space from the 2K Memory Pool 9-27
Moving Space from Other Memory Pools 9-28
Dropping a Memory Pool 9-29

When Pools Cannot Be Dropped Due to Pages In Use 9-29
Cache Binding Effects on Memory and Query Plans 9-30
Flushing Pages from Cache 9-30
Locking to Perform Bindings 9-30
Cache Binding Effects on Stored Procedures and Triggers 9-30
Configuring Data Caches with the Configuration File 9-31
Cache and Pool Entries in the Configuration File 9-31
Configuration File Errors 9-34
Cache Configuration Guidelines 9-35

10. Managing Multiprocessor Servers

Parallel Processing 10-1
Definitions 10-1
Target Architecture 10-2
Adaptive Server Task Management for SMP 10-4
Configuring an SMP Environment 10-5
Managing Engines 10-6
Resetting the Number of Engines 10-6
Choosing the Right Number of Engines 10-7
Monitoring CPU Usage 10-7
Managing User Connections 10-7
Managing Memory 10-8
Configuration Parameters That Affect SMP Systems 10-9
Configuring Spinlock Ratio Parameters 10-9

11. Setting Configuration Parameters

Adaptive Server Configuration Parameters 11-1
What Are Configuration Parameters? 11-5
The Adaptive Server Configuration File 11-6
How to Modify Configuration Parameters 11-6
Who Can Modify Configuration Parameters 11-6
Getting Help Information on Configuration Parameters 11-7
Using *sp_configure* 11-8
Syntax Elements 11-9
Parameter Parsing 11-9
Using *sp_configure* with a Configuration File 11-10
Naming Tips for the Configuration File 11-10
Using *sp_configure* to Read or Write the Configuration File 11-11
Parameters for Using Configuration Files 11-11

Editing the Configuration File by Hand 11-12
Starting Adaptive Server with a Configuration File 11-14
The Parameter Hierarchy 11-15
User-Defined Subsets of the Parameter Hierarchy: Display Levels . . 11-16
The Effect of the Display Level on *sp_configure* Output 11-17
The *reconfigure* Command 11-18
Performance Tuning with *sp_configure* and *sp_sysmon* 11-18
Output from *sp_configure* 11-18
The *sysconfigures* and *syscurconfigs* Tables 11-20
Querying *syscurconfigs* and *sysconfigures*: An Example 11-20
Details on Configuration Parameters 11-21
Backup and Recovery 11-21
number of large i/o buffers 11-21
print recovery information 11-22
recovery interval in minutes 11-22
tape retention in days 11-25
Cache Manager 11-25
global async prefetch limit 11-26
memory alignment boundary 11-26
number of index trips 11-27
number of oam trips 11-28
procedure cache percent 11-29
total data cache size 11-30
Component Integration Services Administration 11-31
cis bulk insert batch size 11-31
cis connect timeout 11-32
cis cursor rows 11-32
cis packet size 11-33
cis rpc handling 11-34
enable cis 11-34
max cis remote connections 11-35
max cis remote servers 11-36
Disk I/O 11-36
allow sql server async i/o 11-36
disk i/o structures 11-37
number of devices 11-38
page utilization percent 11-39
Error Log 11-40
event log computer name (Windows NT only) 11-40
event logging (Windows NT only) 11-41
log audit logon failure 11-42

log audit logon success 11-42
Extended Stored Procedures 11-43
esp execution priority 11-43
esp execution stacksize 11-44
esp unload dll 11-44
start mail session (Windows NT only) 11-45
xp_cmdshell context 11-46
General Information 11-47
configuration file 11-47
Languages 11-47
default character set id 11-48
default language id 11-48
default sortorder id 11-49
disable character set conversions 11-49
number of languages in cache 11-50
Lock Manager 11-50
address lock spinlock ratio 11-51
deadlock checking period 11-52
deadlock retries 11-53
freelock transfer block size 11-55
max engine freelocks 11-56
number of locks 11-58
page lock spinlock ratio 11-59
table lock spinlock ratio 11-60
Memory Use 11-61
executable codesize + overhead 11-61
Metadata Caches 11-61
number of open databases 11-62
number of open indexes 11-64
number of open objects 11-66
open index hash spinlock ratio 11-69
open index spinlock ratio 11-70
open object spinlock ratio 11-71
Network Communication 11-71
allow remote access 11-72
default network packet size 11-72
max network packet size 11-74
max number network listeners 11-77
number of remote connections 11-78
number of remote logins 11-78
number of remote sites 11-79

remote server pre-read packets 11-79
tcp no delay 11-80
O / S Resources 11-81
max async i/os per engine 11-81
max async i/os per server 11-81
o/s file descriptors 11-83
shared memory starting address 11-84
Parallel Queries 11-84
number of worker processes 11-86
max parallel degree 11-86
max scan parallel degree 11-87
memory per worker process 11-88
Physical Memory 11-89
additional network memory 11-89
lock shared memory 11-91
max SQL text monitored 11-91
total memory 11-92
Processors 11-93
max online engines 11-94
Rep Agent Thread Administration 11-94
enable rep agent threads 11-95
SQL Server Administration 11-95
allow backward scans 11-96
allow nested triggers 11-97
allow resource limits 11-97
allow updates to system tables 11-98
cpu accounting flush interval 11-99
cpu grace time 11-100
default database size 11-101
default fill factor percent 11-102
dump on conditions 11-103
event buffers per engine 11-103
housekeeper free write percent 11-104
identity burning set factor 11-106
identity grab size 11-107
i/o accounting flush interval 11-108
i/o polling process count 11-109
lock promotion HWM 11-110
lock promotion LWM 11-111
lock promotion PCT 11-112
maximum dump conditions 11-113

number of alarms 11-113
number of aux scan descriptors 11-114
number of mailboxes 11-117
number of messages. 11-118
number of pre-allocated extents 11-118
number of sort buffers 11-119
partition groups 11-120
partition spinlock ratio 11-121
print deadlock information 11-122
runnable process search count 11-123
size of auto identity column 11-124
SQL Perfmon Integration (Windows NT only). 11-125
sql server clock tick length 11-126
time slice 11-127
upgrade version. 11-128
Security Related 11-128
allow procedure grouping 11-129
auditing. 11-129
audit queue size. 11-130
current audit table. 11-131
max roles enabled per user 11-132
secure default login 11-133
select on syscomments.text column 11-134
suspend audit when device full 11-134
systemwide password expiration 11-135
unified login required (Windows NT Only). 11-136
use security services (Windows NT Only*)*. 11-137
User Environment 11-137
number of user connections 11-137
permission cache entries 11-139
stack guard size 11-140
stack size 11-143
user log cache size 11-144
user log cache spinlock ratio 11-145

12. Limiting Access to Server Resources

What Are Resource Limits? 12-1
Planning Resource Limits 12-2
Enabling Resource Limits 12-2
Defining Time Ranges 12-3

Determining What Time Ranges You Need 12-4
Creating Named Time Ranges 12-4
A Time Range Example 12-5
Modifying a Named Time Range 12-6
Dropping a Named Time Range 12-6
When Do Time Range Changes Take Effect? 12-7
Identifying Users and Limits 12-7
Identifying Heavy-Usage Users 12-8
Identifying Heavy-Usage Applications 12-8
Choosing a Limit Type 12-10
Determining Time of Enforcement 12-11
Determining the Scope of Resource Limits 12-11
Understanding Limit Types 12-13
Limiting I/O Cost 12-13
Identifying I/O Costs 12-14
Calculating the I/O Cost of a Cursor 12-14
The Scope of the *io_cost* Limit Type 12-15
Limiting Elapsed Time 12-15
The Scope of the *elapsed_time* Limit Type 12-16
Limiting the Size of the Result Set 12-16
Determining Row Count Limits 12-16
Applying Row Count Limits to a Cursor 12-17
The Scope of the *row_count* Limit Type 12-17
Creating a Resource Limit 12-17
Resource Limit Examples 12-18
Example 1 12-18
Example 2 12-18
Example 3 12-19
Getting Information on Existing Limits 12-19
Example of Listing All Existing Resource Limits 12-20
Modifying Resource Limits 12-21
Examples of Modifying a Resource Limit 12-22
Dropping Resource Limits 12-22
Examples of Dropping a Resource Limit 12-24
Consequences of Dropping a Login 12-24
Resource Limit Precedence 12-24
Time Ranges 12-25
Resource Limits 12-25

13. Configuring Character Sets, Sort Orders, and Languages

Language Support for International Installations 13-1

Character Sets and Sort Orders 13-2
Types of Internationalization Files 13-2
Character Sets Directory Structure 13-3
Character Sets and Global Variables 13-3
Software Messages 13-4
Types of Localization Files 13-4
Software Messages Directory Structure 13-5
Message Languages and Global Variables 13-6
Disabling Character Set Conversion Between Adaptive Server and Clients 13-6
Changing the Default Character Set, Sort Order, or Language 13-6
Changing the Default Character Set 13-7
Changing the Default Sort Order 13-8
Getting Information About Sort Orders 13-8
Database Dumps and Configuration Changes 13-8
The Steps Involved 13-9
Preliminary Steps 13-9
Steps to Configure Languages, Character Sets, and Sort Orders . . 13-9
Final Steps 13-10
Setting the User's Default Language 13-10
If You Changed the Sort Order or Default Character Set 13-10
Recovery After Reconfiguration 13-11
Using *sp_indsuspect* to Find Corrupt Indexes 13-11
Rebuilding Indexes After Changing the Sort Order 13-12
Upgrading *text* Data After Changing Character Sets 13-13
Retrieving *text* Values After Changing Character Sets 13-14
Installing Date Strings for Unsupported Languages 13-15
Server vs. Client Date Interpretation 13-15

14. Configuring Client/Server Character Set Conversions

Character Set Conversion in Adaptive Server 14-1
Conversion Paths Supported 14-1
Characters That Cannot Be Converted 14-2
Error Handling in Character Set Conversion 14-3
Setting Up the Conversion Process 14-4
Specifying the Character Set for Utility Programs 14-5
Controlling Character Conversion During a Session 14-6
Display and File Character Set Command Line Options 14-7
Setting the Display Character Set 14-8
Setting the File Character Set 14-8

15. Creating and Managing User Databases

Commands for Creating and Managing User Databases 15-1
Permissions for Managing User Databases 15-2
Using the *create database* Command 15-3
create database Syntax 15-3
How *create database* Works 15-4
Adding Users to Databases 15-5
Assigning Space and Devices to Databases 15-5
Default Database Size and Devices 15-6
Estimating the Required Space 15-7
Placing the Transaction Log on a Separate Device 15-7
Estimating the Transaction Log Size 15-8
Default Log Size and Device 15-9
Moving the Transaction Log to Another Device 15-9
Using the *for load* Option for Database Recovery 15-10
Using the *with override* Option with *create database* 15-11
Changing Database Ownership 15-12
Using the *alter database* Command 15-12
alter database Syntax 15-13
Using the *with override* Clause 15-14
Using the *for load* Clause 15-14
Using the *drop database* Command 15-14
drop database Syntax 15-15
System Tables That Manage Space Allocation 15-15
The *sysusages* Table 15-16
The *segmap* Column 15-17
The *lstart*, *vstart*, and *size* Columns 15-17
Getting Information About Database Storage 15-18
Database Device Names and Options 15-18
Checking the Amount of Space Used 15-19
Checking Space Used in a Database 15-19
Checking Summary Information for a Table 15-20
Checking Information for a Table and Its Indexes 15-21
Querying System Table for Space Usage Information 15-21

16. Setting Database Options

What Are Database Options? 16-1
Using the *sp_dboption* Procedure 16-1
Database Option Descriptions 16-2
abort tran on log full 16-2

allow nulls by default 16-3
auto identity 16-3
dbo use only 16-3
ddl in tran 16-3
identity in nonunique index 16-5
no chkpt on recovery 16-5
no free space acctg 16-6
read only 16-6
select into/bulkcopy/pllsort 16-6
single user 16-7
trunc log on chkpt 16-7
unique auto_identity index 16-8
Changing Database Options 16-8
Viewing the Options on a Database 16-9

17. Creating and Using Segments

What Is a Segment? 17-1
System-Defined Segments 17-2
Commands and Procedures for Managing Segments 17-3
Why Use Segments? 17-3
Controlling Space Usage 17-4
Segments and Thresholds 17-4
Improving Performance 17-4
Separating Tables, Indexes, and Logs 17-5
Splitting Tables 17-5
Moving a Table to Another Device 17-7
Creating Segments 17-7
Changing the Scope of Segments 17-8
Extending the Scope of Segments 17-8
Automatically Extending the Scope of a Segment 17-8
Reducing the Scope of a Segment 17-9
Assigning Database Objects to Segments 17-10
Creating New Objects on Segments 17-10
Placing Existing Objects on Segments 17-12
Placing Text Pages on a Separate Device 17-15
Creating Clustered Indexes on Segments 17-15
Dropping Segments 17-16
Getting Information About Segments 17-16
sp_helpsegment 17-17
sp_helpdb 17-18

sp_help and *sp_helpindex* 17-18
Segments and System Tables 17-18
A Segment Tutorial 17-19
Segments and Clustered Indexes 17-24

18. Checking Database Consistency

What Is the Database Consistency Checker? 18-1
Understanding Page and Object Allocation Concepts 18-2
Understanding the Object Allocation Map (OAM) 18-4
Understanding Page Linkage 18-5
What Checks Can Be Performed with *dbcc*? 18-6
Checking Consistency of Databases and Tables 18-7
dbcc checkstorage 18-7
Advantages of Using *dbcc checkstorage* 18-8
Comparison of *dbcc checkstorage* and Other *dbcc* Commands 18-8
Understanding the *dbcc checkstorage* Operation 18-9
Performance and Scalability 18-10
Consistency Checking with *dbcc checkstorage* 18-10
dbcc checktable 18-11
dbcc checkdb 18-13
Checking Page Allocation 18-13
dbcc checkalloc 18-13
dbcc indexalloc 18-15
dbcc tablealloc 18-16
Correcting Allocation Errors Using the *fix* | *nofix* Option 18-16
Generating Reports with *dbcc tablealloc* and *dbcc indexalloc* 18-17
Checking Consistency of System Tables 18-18
Strategies for Using Consistency Checking Commands 18-18
Comparing the Performance of *dbcc* Commands 18-19
Using Large I/O and Asynchronous Prefetch 18-20
Scheduling Database Maintenance at Your Site 18-20
Database Use 18-21
Backup Schedule 18-22
Size of Tables and Importance of Data 18-22
Understanding the Output from *dbcc* commands 18-23
Errors Generated by Database Consistency Problems 18-24
Comparison of Soft and Hard Faults 18-25
Soft Faults 18-25
Hard Faults 18-26
Dropping a Damaged Database 18-26

Preparing to Use *dbcc checkstorage* 18-27
Planning Resources 18-28
Examples of *sp_plan_dbccdb* Output 18-28
Planning Workspace Size 18-30
Configuring Adaptive Server for *dbcc checkstorage* 18-31
Configuring Worker Processes 18-31
Setting Up a *dbcc* Named Cache. 18-33
Configuring a 16K I/O buffer pool 18-34
Allocating Disk Space for *dbccdb* 18-34
Segments for Workspaces 18-35
Creating the *dbccdb* Database 18-35
Updating the *dbcc_config* Table 18-38
Maintaining *dbccdb* 18-38
Reevaluating and Updating *dbccdb* Configuration 18-39
Cleaning Up *dbccdb* 18-40
Performing Consistency Checks on *dbccdb* 18-40
Generating Reports from *dbccdb* 18-41
To Report a Summary of *dbcc checkstorage* Operations 18-41
To Report Configuration, Statistics and Fault Information 18-41
To See Configuration Information for a Target Database 18-42
To Compare Results of *dbcc checkstorage* Operations 18-42
To Report Faults Found in a Database Object. 18-43
To Report Statistics Information from *dbcc_counter* 18-43

19. *dbccdb* Tables

dbcc_config 19-1
dbcc_counters 19-2
dbcc_fault_params 19-3
dbcc_faults 19-3
dbcc_operation_log 19-4
dbcc_operation_results 19-5
dbcc_types 19-6
***dbccdb* Workspaces** 19-13
***dbccdb* Log** 19-15

20. Developing a Backup and Recovery Plan

Keeping Track of Database Changes 20-2
Getting Information About the Transaction Log 20-3
Synchronizing a Database and Its Log: Checkpoints 20-3

Setting the Recovery Interval 20-3
Automatic Checkpoint Procedure 20-4
Checkpoint After User Database Upgrade 20-4
Truncating the Log After Automatic Checkpoints 20-4
Free Checkpoints 20-5
Manually Requesting a Checkpoint 20-6
Automatic Recovery After a System Failure or Shutdown 20-6
Determining Whether Messages Are Displayed During Recovery 20-7
Fault Isolation During Recovery 20-7
Persistence of Offline Pages 20-8
Configuring Recovery Fault Isolation 20-8
Isolating Suspect Pages 20-8
Raising the Number of Suspect Pages Allowed 20-9
Getting Information About Offline Databases and Pages 20-10
Bringing Offline Pages Online 20-11
Side Effects of Offline Pages 20-12
Recovery Strategies Using Recovery Fault Isolation 20-13
Reload Strategy 20-14
Repair Strategy 20-15
Assessing the Extent of Corruption 20-16
Using the Dump and Load Commands 20-17
Checking Database Consistency: *dbcc* 20-17
Making Routine Database Dumps: *dump database* 20-18
Making Routine Transaction Log Dumps: *dump transaction* 20-18
Copying the Log After Device Failure: *dump tran with no_truncate* . . . 20-18
Restoring the Entire Database: *load database* 20-19
Applying Changes to the Database: *load transaction* 20-19
Making the Database Available to Users: *online database* 20-20
Moving a Database to Another Adaptive Server 20-20
Upgrading a Single-User Database 20-21
Using the Special *dump transaction* Options 20-22
Using the Special Load Options to Identify Dump Files 20-22
Restoring a Database from Backups 20-23
Designating Responsibility for Backups 20-25
Using the Backup Server for Backup and Recovery 20-25
Relationship Between Adaptive Server and Backup Servers 20-26
Communicating with the Backup Server 20-28
Mounting a New Volume 20-28
Starting and Stopping Backup Server 20-30
Configuring Your Server for Remote Access 20-30
Choosing Backup Media 20-31

Protecting Backup Tapes from Being Overwritten 20-31
Dumping to Files or Disks..... 20-31
Creating Logical Device Names for Local Dump Devices..... 20-32
Listing the Current Device Names..... 20-32
Adding a Backup Device..... 20-33
Redefining a Logical Device Name 20-33
Scheduling Backups of User Databases 20-33
Scheduling Routine Backups 20-34
Other Times to Back Up a Database..... 20-34
Dumping a User Database After Upgrading..... 20-34
Dumping a Database After Creating an Index 20-34
Dumping a Database After Unlogged Operations 20-35
Dumping a Database When the Log Has Been Truncated..... 20-35
Scheduling Backups of *master* 20-35
Dumping *master* After Each Change 20-36
Saving Scripts and System Tables..... 20-36
Truncating the *master* Database Transaction Log..... 20-37
Avoiding Volume Changes and Recovery 20-37
Scheduling Backups of the *model* Database 20-37
Truncating the *model* Database's Transaction Log 20-37
Scheduling Backups of the *sybsystemprocs* Database 20-38
Configuring Adaptive Server for Simultaneous Loads..... 20-38
Gathering Backup Statistics 20-39

21. Backing Up and Restoring User Databases

Dump and Load Command Syntax 21-2
Specifying the Database and Dump Device 21-5
Rules for Specifying Database Names..... 21-6
Rules for Specifying Dump Devices 21-7
Examples 21-7
Tape Device Determination by Backup Server..... 21-8
Tape Device Configuration File 21-9
Specifying a Remote Backup Server 21-10
Specifying Tape Density, Block Size, and Capacity 21-11
Overriding the Default Density 21-12
Overriding the Default Block Size 21-13
Specifying Tape Capacity for Dump Commands 21-13
Specifying the Volume Name..... 21-14
Loading from a Multifile Volume..... 21-15
Identifying a Dump 21-15

Specifying Additional Dump Devices: the *stripe on* Clause 21-17
Dumping to Multiple Devices 21-18
Loading from Multiple Devices 21-19
Using Fewer Devices to Load Than to Dump 21-19
Specifying the Characteristics of Individual Devices 21-20
Tape Handling Options 21-20
Specifying Whether to Dismount the Tape 21-21
Rewinding the Tape 21-21
Protecting Dump Files from Being Overwritten 21-22
Reinitializing a Volume Before a Dump 21-22
Dumping Multiple Databases to a Single Volume 21-23
Overriding the Default Message Destination 21-24
Getting Information About Dump Files 21-25
Requesting Dump Header Information 21-26
Determining the Database, Device, File Name, and Date 21-27
Copying the Log After a Device Failure 21-28
Truncating a Log That Is Not on a Separate Segment 21-30
Truncating the Log in Early Development Environments 21-30
Truncating a Log That Has No Free Space 21-31
Dangers of Using *with truncate_only* and *with no_log* 21-31
Providing Enough Log Space 21-32
The *syslogshold* Table 21-32
Responding to Volume Change Requests 21-34
sp_volchanged Syntax 21-35
Volume Change Prompts for Dumps 21-35
Volume Change Prompts for Loads 21-38
Recovering a Database: Step-by-Step Instructions 21-39
Getting a Current Dump of the Transaction Log 21-40
Examining the Space Usage 21-40
Dropping the Databases 21-42
Dropping the Failed Devices 21-42
Initializing New Devices 21-42
Re-Creating the Databases 21-43
Loading the Database 21-44
Loading the Transaction Logs 21-44
Loading a Transaction Log to a Point in Time 21-44
Bringing the Databases Online 21-45
Replicated Databases 21-45
Upgrading User Database Dumps 21-46
How to Upgrade a Dump to Adaptive Server Release 11.5.x 21-47
The “Database Offline” Status Bit 21-48

Version Identifiers 21-49
Configuration Upgrade Version ID 21-49
Upgrade Version Indicator 21-49
Log Compatibility Version Specifier 21-49
Cache Bindings and Loading Databases 21-49
Databases and Cache Bindings 21-50
Database Objects and Cache Bindings 21-50
Checking on Cache Bindings 21-51
Cross-Database Constraints and Loading Databases 21-51

22. Restoring the System Databases

What Does Recovering a System Database Entail? 22-1
Symptoms of a Damaged *master* Database 22-2
Recovering the *master* Database 22-2
About the Recovery Process 22-3
Summary of Recovery Procedure 22-3
Step One: Find Copies of System Tables 22-4
Step Two: Build a New Master Device 22-5
Step Three: Start Adaptive Server in Master-Recover Mode 22-6
Step Four: Re-Create Device Allocations for *master* 22-7
Determining Which Allocations Are on the Master Device 22-7
Creating Additional Allocations 22-9
Step Five: Check Your Backup Server *sysservers* Information 22-12
Step Six: Verify That Your Backup Server Is Running 22-13
Step Seven: Load a Backup of *master* 22-13
Step Eight: Update the *number of devices* Configuration Parameter . . . 22-13
Step Nine: Restart Adaptive Server in Master-Recover Mode 22-14
Step Ten: Check System Tables to Verify Current Backup of *master* . . 22-14
Step Eleven: Restart Adaptive Server 22-15
Step Twelve: Restore Server User IDs 22-15
Step Thirteen: Restore the *model* Database 22-15
Step Fourteen: Check Adaptive Server 22-16
Step Fifteen: Back Up *master* 22-16
Recovering the *model* Database 22-17
Restoring the Generic *model* Database 22-17
Restoring *model* from a Backup 22-17
Restoring *model* with No Backup 22-18
Recovering the *sybsystemprocs* Database 22-18
Restoring *sybsystemprocs* with *installmaster* 22-18
Restoring *sybsystemprocs* with *load database* 22-20

Restoring System Tables with *disk reinit* and *disk refit* 22-21
Restoring *sysdevices* with *disk reinit* 22-21
Restoring *sysusages* and *sysdatabase* with *disk refit* 22-22

23. Managing Free Space with Thresholds

Monitoring Free Space with the Last-Chance Threshold 23-1
Crossing the Threshold 23-2
Controlling How Often *sp_thresholdaction* Executes 23-3
Aborting or Suspending Processes 23-4
Aborting Processes 23-4
Waking Suspended Processes 23-4
Adding, Changing, and Deleting Thresholds 23-5
Displaying Information About Existing Thresholds 23-5
Thresholds and System Tables 23-5
Adding a Free-Space Threshold 23-6
Changing a Free-Space Threshold 23-6
Specifying a New Last-Chance Threshold Procedure 23-7
Dropping a Threshold 23-7
Creating a Free-Space Threshold for the Log Segment 23-8
Adding a Log Threshold at 45 Percent of Log Size 23-8
Testing and Adjusting the New Threshold 23-9
Creating Additional Thresholds on Other Segments 23-11
Determining Threshold Placement 23-11
Creating Threshold Procedures 23-12
Declaring Procedure Parameters 23-13
Generating Error Log Messages 23-13
Dumping the Transaction Log 23-13
A Simple Threshold Procedure 23-14
A More Complex Procedure 23-15
Deciding Where to Put a Threshold Procedure 23-17
Disabling Free-Space Accounting for Data Segments 23-18

Index

List of Figures

Figure 1-1: Connecting to Adaptive Server 1-11
Figure 4-1: Error log format 4-6
Figure 4-2: How SQL text is truncated if not enough memory is configured 4-19
Figure 5-1: System tables that manage storage 5-7
Figure 7-1: Disk mirroring using minimal physical disk space 7-3
Figure 7-2: Disk mirroring for rapid recovery 7-4
Figure 7-3: Disk mirroring: keeping transaction logs on a separate disk 7-5
Figure 8-1: Example of memory allocation 8-3
Figure 8-2: How changing configuration parameters reduces cache size 8-4
Figure 9-1: Data cache with default cache and two named data caches 9-3
Figure 9-2: Configuring a cache and a 4K memory pool 9-12
Figure 9-3: Moving space from an existing pool to a new pool 9-12
Figure 9-4: Wash area of a buffer pool 9-21
Figure 9-5: Small wash area results in a dirty buffer grab 9-22
Figure 9-6: Effects of making the wash area too large 9-23
Figure 10-1: SMP environment architecture 10-3
Figure 10-2: Adaptive Server task management in the SMP environment 10-5
Figure 10-3: Relationship between spinlocks and index descriptors 10-10
Figure 11-1: The checkpoint process 11-24
Figure 11-2: Deadlocks during page splitting in a clustered index 11-54
Figure 11-3: Factors in determining packet size 11-76
Figure 11-4: Precedence of parallel options 11-85
Figure 11-5: Process about to corrupt stack guardword 11-141
Figure 13-1: Messages directory structure 13-5
Figure 14-1: Comparison of EUC JIS and Shift-JIS encoding for Japanese characters 14-3
Figure 14-2: Where character set conversion may be needed 14-7
Figure 17-1: System-defined segments 17-2
Figure 17-2: Partitioning a table across physical devices 17-5
Figure 17-3: Creating objects on specific devices using segments 17-11
Figure 17-4: Splitting a large table across two segments 17-14
Figure 18-1: Page management with extents 18-3
Figure 18-2: OAM page and allocation page pointers 18-5
Figure 18-3: How a newly allocated page is linked with other pages 18-6
Figure 20-1: Reload strategy 20-15
Figure 20-2: Repair strategy 20-16
Figure 20-3: Restoring a database, a scenario 20-23
Figure 20-4: Restoring a database, a second scenario 20-24

Figure 20-5: Adaptive Server and Backup Server with remote Backup Server....20-27
Figure 21-1: File-naming convention for database and transaction log dumps....21-17
Figure 21-2: Dumping several databases to the same volume....21-24
Figure 22-1: Determining allocations on the master device....22-8
Figure 22-2: Sample output from sysusages....22-8
Figure 22-3: Allocations on a master device....22-9
Figure 22-4: Sample sysusages output with additional allocations....22-10
Figure 22-5: Complex allocations on a master device....22-11
Figure 23-1: Log segment with a last-chance threshold....23-2
Figure 23-2: Executing sp_thresholdaction when the last-chance threshold is reached....23-3
Figure 23-3: Free space must rise by @@thresh_hysteresis to reactivate threshold....23-3
Figure 23-4: Transaction log with additional threshold at 45 percent....23-9
Figure 23-5: Moving threshold leaves less free space after dump....23-10
Figure 23-6: Additional log threshold does not begin dump early enough....23-10
Figure 23-7: Moving threshold leaves enough free space to complete dump....23-11
Figure 23-8: Determining where to place a threshold....23-12

List of Tables

Table 1: Syntax statement conventions xxxvi
Table 2: Types of expressions used in syntax statements xxxviii
Table 2-1: Information the master database tracks 2-2
Table 4-1: Error text symbols key 4-3
Table 4-2: Status values reported by sp_who 4-15
Table 4-3: SQL commands not represented by text 4-21
Table 4-4: Columns added to sysprocesses 4-23
Table 5-1: Device allocation topics 5-1
Table 5-2: Object placement topics 5-2
Table 5-3: Commands for allocating disk resources 5-2
Table 5-4: Commands for placing objects on disk resources 5-3
Table 6-1: Status bits in sysdevices 6-7
Table 7-1: Effects of mode and side options to the disk mirror command 7-9
Table 9-1: Procedures and commands for using named caches 9-3
Table 10-1: Spinlock ratio configuration parameters 10-9
Table 11-1: sp_configure syntax 11-8
Table 12-1: Resource limit types 12-10
Table 12-2: Values for *sp_help_resource_limit* output 12-20
Table 12-3: Identifying resource limits to drop 12-23
Table 13-1: Internationalization files 13-2
Table 13-2: Localization files 13-4
Table 14-1: Supported character set conversions 14-2
Table 15-1: Commands for managing user databases 15-1
Table 15-2: Columns in sp_spaceused output 15-20
Table 17-1: System-defined segments 17-2
Table 17-2: Commands and procedures for managing segments 17-3
Table 18-1: Comparison of checks performed by *dbcc* commands 18-6
Table 18-2: Comparison of the performance of dbcc commands 18-19
Table 18-3: Tasks for preparing to use *dbcc checkstorage* 18-27
Table 19-1: Columns in the dbcc_config table 19-1
Table 19-2: Columns in the dbcc_counters table 19-2
Table 19-3: Columns in the dbcc_fault_params table 19-3
Table 19-4: Columns in the dbcc_faults table 19-3
Table 19-5: Columns in the dbcc_operation_log table 19-4
Table 19-6: Columns in the dbcc_operation_results table 19-5
Table 19-7: Contents of the dbcc_types table 19-6
Table 20-1: Further information about backup and recovery 20-2

Table 20-2: When to use dump transaction with truncate_only or with no_log 20-22
Table 20-3: Changing tape volumes on a UNIX system 20-29
Table 21-1: Further information about backup and recovery 21-2
Table 21-2: Syntax for routine dumps and log dumps after device failure 21-3
Table 21-3: Syntax for load commands 21-4
Table 21-4: Special dump transaction options 21-5
Table 21-5: Indicating the database name and dump device 21-6
Table 21-6: Dumping to or loading from a remote Backup Server 21-10
Table 21-7: Specifying tape density, block size, and capacity 21-12
Table 21-8: Specifying the volume name 21-14
Table 21-9: Specifying the file name for a dump 21-16
Table 21-10: Using more than one dump device 21-18
Table 21-11: Tape handling options 21-21
Table 21-12: Overriding the default message destination 21-25
Table 21-13: Listing dump headers or file names 21-26
Table 21-14: Copying the log file after a device failure 21-29
Table 21-15: Sample device allocation 21-41
Table 22-1: Further information about backup and recovery 22-1
Table 22-2: Using sysdevices to determine disk reinit parameters 22-21
Table 23-1: Further information about backup and recovery 23-1

About This Book

This manual, the *Sybase Adaptive Server System Administration Guide*, describes how to administer and control Sybase® Adaptive Server™ Enterprise databases independent of any specific database application.

Audience

This manual is intended for Sybase System Administrators and Database Owners.

How to Use This Book

This manual contains five sections. Part 1, “Introduction,” describes basic system administration issues:

- Chapter 1, “Overview of System Administration,” describes the structure of the Sybase system.
- Chapter 2, “System Databases,” discusses the contents and function of the Adaptive Server system databases.
- Chapter 3, “System Administration for Beginners,” summarizes important tasks that new System Administrators need to perform.
- Chapter 4, “Diagnosing System Problems,” discusses Adaptive Server and Backup Server™ error handling and shows how to shut down servers and kill user processes.

Part 2, “Managing Physical Resources,” describes how to set up and use disks, memory, and processors with Adaptive Server:

- Chapter 5, “Overview of Disk Resource Issues,” provides an overview of Adaptive Server disk resource issues.
- Chapter 6, “Initializing Database Devices,” describes how to initialize and use database devices.
- Chapter 7, “Mirroring Database Devices,” describes how to mirror database devices for nonstop recovery from media failures.
- Chapter 8, “Configuring Memory,” explains how to configure Adaptive Server to use the available memory on your system.

- Chapter 9, “Configuring Data Caches,” discusses how to create named caches in memory and bind objects to those caches.
- Chapter 10, “Managing Multiprocessor Servers,” explains how to use multiple CPUs with Adaptive Server and discusses system administration issues that are unique to symmetric multiprocessing (SMP) environments.

Part 3, “Configuring Adaptive Server Behavior,” explains how to configure and use different Adaptive Server features:

- Chapter 11, “Setting Configuration Parameters,” summarizes the configuration parameters that you set with **sp_configure**, which control many aspects of Adaptive Server behavior.
- Chapter 12, “Limiting Access to Server Resources,” explains how to create and manage resource limits with Adaptive Server.
- Chapter 13, “Configuring Character Sets, Sort Orders, and Languages,” discusses international issues, such as the files included in the Language Modules and how to configure an Adaptive Server language, sort order, and character set.
- Chapter 14, “Configuring Client / Server Character Set Conversions,” discusses character set conversion between Adaptive Server and clients in a heterogeneous environment.

Part 4, “Managing Databases and Database Objects,” describes how to create and administer databases and segments:

- Chapter 15, “Creating and Managing User Databases,” discusses the physical placement of databases, tables, and indexes, and the allocation of space to them.
- Chapter 16, “Setting Database Options,” describes how to set database options.
- Chapter 17, “Creating and Using Segments,” describes how to use segments, which are named collections of database devices, in databases.
- Chapter 18, “Checking Database Consistency,” describes how to use the database consistency checker, **dbcc**, to detect and fix database problems.
- Chapter 19, “dbccdb Tables,” describes the tables that reside in the *dbccdb* databases, which you must install in order to use certain **dbcc** commands.

Part 5, “Backup and Recovery,” describes how to develop and execute a backup and recovery plan for the Adaptive Server system.

- Chapter 20, “Developing a Backup and Recovery Plan,” discusses the capabilities of the Backup Server and how to develop your backup strategy.
- Chapter 21, “Backing Up and Restoring User Databases,” discusses how to recover user databases.
- Chapter 22, “Restoring the System Databases,” discusses how to recover system databases.
- Chapter 23, “Managing Free Space with Thresholds,” discusses managing space with thresholds.

Adaptive Server Enterprise Documents

The following documents comprise the Sybase Adaptive Server Enterprise documentation:

- The *Release Bulletin* for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the *Release Bulletin* may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use SyBooks™-on-the-Web.
- The Adaptive Server installation documentation for your platform – describes installation and upgrade procedures for all Adaptive Server and related Sybase products.
- The Adaptive Server configuration documentation for your platform – describes configuring a server, creating network connections, configuring for optional functionality, such as auditing, installing most optional system databases, and performing operating system administration tasks.
- *What’s New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server release 11.5, the system changes added to support those features, and the changes that may affect your existing applications.
- *Navigating the Documentation for Adaptive Server* – an electronic interface for using Adaptive Server. This online document provides links to the concepts and syntax in the documentation that are relevant to each task.
- *Transact-SQL User’s Guide* – documents Transact-SQL, Sybase’s enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database

management system. This manual also contains descriptions of the *pubs2* and *pubs3* sample databases.

- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources and user and system databases, and specifying character conversion, international language, and sort order settings.
- *Adaptive Server Reference Manual* – contains detailed information about all Transact-SQL commands, functions, procedures, and datatypes. This manual also contains a list of the Transact-SQL reserved words and definitions of system tables.
- *Performance and Tuning Guide* – explains how to tune Adaptive Server for maximum performance. This manual includes information about database design issues that affect performance, query optimization, how to tune Adaptive Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.
- The *Utility Programs* manual for your platform – documents the Adaptive Server utility programs, such as **isql** and **bcp**, which are executed at the operating system level.
- *Security Administration Guide* – explains how to use the security features provided by Adaptive Server to control user access to data. This manual includes information about how to add users to Adaptive Server, administer both system and user-defined roles, grant database access to users, and manage remote Adaptive Servers.
- *Security Features User's Guide* – provides instructions and guidelines for using the security options provided in Adaptive Server from the perspective of the non-administrative user.
- *Error Messages* and *Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Component Integration Services User's Guide for Adaptive Server Enterprise and OmniConnect* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Adaptive Server Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Master Index for Adaptive Server Publications* – combines the indexes of the *Adaptive Server Reference Manual*, *Component*

Integration Services User's Guide, *Performance and Tuning Guide*, *Security Administration Guide*, *Security Features User's Guide*, *System Administration Guide*, and *Transact-SQL User's Guide*.

Other Sources of Information

Use the SyBooks™ and SyBooks-on-the-Web online resources to learn more about your product:

- SyBooks documentation is on the CD that comes with your software. The DynaText browser, also included on the CD, allows you to access technical information about your product in an easy-to-use format.

  Refer to *Installing SyBooks* in your documentation package for instructions on installing and starting SyBooks.

- SyBooks-on-the-Web is an HTML version of SyBooks that you can access using a standard Web browser.

  To use SyBooks-on-the-Web, go to http://www.sybase.com, and choose Documentation.

Conventions Used in This Manual

The following sections describe the style conventions used in this manual.

Formatting SQL Statements

SQL is a free-form language: there are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented.

SQL Syntax Conventions

The conventions for syntax statements in this manual are as follows:

Table 1: Syntax statement conventions

Key	Definition
`command`	Command names, command option names, utility names, utility flags, and other keywords are in **`bold Courier`** in syntax statements, and in **bold Helvetica** in paragraph text.
`variable`	Variables, or words that stand for values that you fill in, are in italics.
`{ }`	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
`[ ]`	Square brackets mean choosing one or more of the enclosed options is optional. Do not include brackets in your option.
`( )`	Parentheses are to be typed as part of the command.
`|`	The vertical bar means you may select only one of the options shown.
`,`	The comma means you may choose as many of the options shown as you like, separating your choices with commas to be typed as part of the command.

- Syntax statements (displaying the syntax and all options for a command) are printed like this:

  ```
  sp_dropdevice [device_name]
  ```

  or, for a command with more options:

  ```
  select column_name
      from table_name
      where search_conditions
  ```

  In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase: normal font for keywords, italics for user-supplied words.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```

- Examples of output from the computer are printed like this:

  ```
  pub_id  pub_name                city        state
  ------- -------------------     ----------- -----
  0736    New Age Books           Boston      MA
  0877    Binnet & Hardley        Washington  DC
  1389    Algodata Infosystems    Berkeley    CA

  (3 rows affected)
  ```

Case

You can disregard case when you type keywords:

SELECT is the same as **Select** is the same as **select**.

Obligatory Options {You Must Choose At Least One}

- **Curly Braces and Vertical Bars:** Choose **one and only one** option.

  ```
  {die_on_your_feet | live_on_your_knees |
  live_on_your_feet}
  ```

- **Curly Braces and Commas:** Choose one or more options. If you choose more than one, separate your choices with commas.

  ```
  {cash, check, credit}
  ```

Optional Options [You Don't Have to Choose Any]

- **One Item in Square Brackets:** You don't have to choose it.

```
[anchovies]
```

- **Square Brackets and Vertical Bars:** Choose **none or only one**.

  ```
  [beans | rice | sweet_potatoes]
  ```

- **Square Brackets and Commas:** Choose **none, one, or more than one** option. If you choose more than one, separate your choices with commas.

  ```
  [extra_cheese, avocados, sour_cream]
  ```

Ellipsis: Do it Again (and Again)...

An ellipsis (. . .) means that you can **repeat** the last unit as many times as you like. In this syntax statement, **buy** is a required keyword:

```
buy thing = price [cash | check | credit]
    [, thing = price [cash | check | credit]]...
```

You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

An ellipsis can also be used inline to signify portions of a command that are left out of a text example. The following syntax statement represents the complete **create database** command, even though required keywords and other options are missing:

```
create database...for load
```

Expressions

Several different types of **expressions** are used in Adaptive Server syntax statements.

Table 2: Types of expressions used in syntax statements

Usage	Definition
expression	Can include constants, literals, functions, column identifiers, variables, or parameters
logical expression	An expression that returns TRUE, FALSE, or UNKNOWN
constant expression	An expression that always returns the same value, such as “5+3” or “ABCDE”

Table 2: Types of expressions used in syntax statements (continued)

Usage	Definition
float_expr	Any floating-point expression or expression that implicitly converts to a floating value
integer_expr	Any integer expression or an expression that implicitly converts to an integer value
numeric_expr	Any numeric expression that returns a single value
char_expr	An expression that returns a single character-type value
binary_expression	An expression that returns a single *binary* or *varbinary* value

If You Need Help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

Introduction

1 Overview of System Administration

This chapter introduces the basic topics of Adaptive Server system administration. Topics include:

- Adaptive Server Database Administration Tasks 1-1
- System Tables 1-4
- System Procedures 1-7
- Logging Error Messages 1-10
- Connecting to Adaptive Server 1-11
- Using Sybase Central for System Administration Tasks 1-3

Adaptive Server Database Administration Tasks

Administering Adaptive Server databases includes tasks such as:

- Installing Adaptive Server and Backup Server
- Granting roles and permissions to Adaptive Server users
- Managing and monitoring the use of disk space, memory, and connections
- Backing up and restoring databases
- Diagnosing system problems
- Configuring Adaptive Server to achieve the best performance

In addition, System Administrators may have a hand in certain database design tasks, such as enforcing integrity standards. This function may overlap with the work of application designers.

Although a System Administrator concentrates on tasks that are independent of the applications running on Adaptive Server, she or he is likely to be the person with the best overview of all the applications. For this reason, a System Administrator can advise application designers about the data that already exists on Adaptive Server, make recommendations about standardizing data definitions across applications, and so on.

However, the distinction between what is and what is not specific to an application is sometimes a bit "fuzzy." Owners of user databases will consult certain sections of this book. Similarly, System Administrators and Database Owners will use the *Transact-SQL User's Guide* (especially the chapters on data definition, stored

procedures, and triggers). Both System Administrators and application designers will use the *Performance and Tuning Guide*.

Roles Required for System Administration Tasks

Many of the commands and procedures discussed in this manual require the System Administrator or System Security Officer role. Other sections in this manual are relevant to Database Owners. A Database Owner's user name within the database is "dbo". You cannot log in as "dbo": a Database Owner logs in under his or her Adaptive Server login name and is recognized as "dbo" by Adaptive Server only while he or she is using the database.

For a complete description of the roles supported by Adaptive Server (System Administrator, System Security Officer, and Operator), see the *Security Administration Guide*. The *Security Administration Guide* also describes the special status of the owners of databases and other objects.

Using *isql* to Perform System Administration Tasks

This book assumes that you will perform the system administration tasks described in this guide by using the command-line utility **isql**. This section provides some basic information about using **isql**. For more information about **isql**, see the *Utility Programs* manual for your platform.

You can also use the graphic tool Sybase Central™ to perform many of the tasks described in this book, as described in "Using Sybase Central for System Administration Tasks" on page 1-3.

Starting *isql*

To start **isql** on most platforms, type this command at an operating system prompt:

```
isql -Uusername
```

where *username* is the user name of the System Administrator. Adaptive Server prompts you for your password.

➤ ***Note***

Do not use the **-P** option of **isql** to specify your password because another user might then see your password.

You can use **isql** in command-line mode to enter many of the Transact-SQL examples in this manual.

Entering Statements

The statements that you enter in **isql** can span several lines. **isql** does not process statements until you type "go" on a separate line. For example:

```
1> select *
2> from sysobjects
3> where type = "TR"
4> go
```

The examples printed in this manual do not include the **go** command between statements. If you are typing the examples, you must enter the **go** command to see the sample output.

Saving and Reusing Statements

This manual frequently asks you to save the Transact-SQL statements you use to create or modify user databases and database objects. The easiest way to accomplish this is to create or copy the statements to an ASCII-formatted file. You can then use the file to supply statements to **isql** if you need to re-create databases or database objects later.

The syntax for using **isql** with an ASCII-formatted file is:

```
isql -Uusername -ifilename
```

where *filename* is the full path and file name of the file that contains Transact-SQL statements. On UNIX and other platforms, use the less than symbol (**<**) to redirect the file.

The Transact-SQL statements in the ASCII file must use valid syntax and the **go** command.

Using Sybase Central for System Administration Tasks

You can accomplish many of the system administration tasks detailed in this book with Sybase Central, a graphic tool that comes with Adaptive Server.

Examples of some of the tasks you can accomplish with Sybase Central are:

- Initializing database devices (Windows NT servers only)

- Setting configuration parameters
- Viewing the amount of free log space in a database
- Generating data definition language (DDL)
- Creating logins
- Adding remote servers
- Creating databases
- Creating stored procedures
- Defining roles
- Adding data caches
- Setting database options
- Backing up and restoring databases

You can also use the Monitor Viewer feature of Sybase Central to access Adaptive Server Monitor™. For more information on using Sybase Central, see *Managing and Monitoring Sybase Adaptive Server Enterprise* in the SyBooks Adaptive Server Enterprise Monitor collection. Sybase Central also comes with extensive online help.

Sybase Central and Transact-SQL Scripts

For recovery reasons, you need to be able to reproduce important system administration work, such as the creation of database objects. Most users do this by saving the work in Transact-SQL scripts, which can be easily rerun using **isql**.

If you are using Sybase Central for system administration tasks, you can use the Sybase Central DDL-generation feature to record your work to Transact-SQL scripts. The DDL-generation feature lets you save to a script the actions you performed in an entire server or within a specific database.

System Tables

The *master* database contains **system tables** that keep track of information about Adaptive Server as a whole. In addition, each database (including the *master* database) contains system tables that keep track of information specific to that database.

All the Adaptive Server-supplied tables in the *master* database (Adaptive Server's controlling database) are considered system tables. Each user database is created with a subset of these system

tables. The system tables may also be referred to as the **data dictionary** or the system catalogs.

A *master* database and its tables are created when Adaptive Server is installed. The system tables in a user database are created when the **create database** command is issued. The names of all system tables start with "sys". You cannot create tables in user databases that have the same names as system tables. An explanation of the system tables and their columns is included in the *Adaptive Server Reference Manual*.

Querying the System Tables

You query system tables just like any other tables. For example, the following statement returns the names of all the triggers in the database:

```
select name
from sysobjects
where type = "TR"
```

In addition, Adaptive Server supplies **stored procedures** (called **system procedures**), many of which provide shortcuts for querying the system tables.

Following is a list of the system procedures that provide information from the system tables:

- **sp_commonkey**
- **sp_configure**
- **sp_countmedatada**
- **sp_dboption**
- **sp_estspace**
- **sp_help**
- **sp_helpartition**
- **sp_helpcache**
- **sp_helpconfig**
- **sp_helpconstraint**
- **sp_helpdb**
- **sp_helpdevice**
- **sp_helpgroup**
- **sp_helpindex**
- **sp_help_resource_limit**
- **sp_helprotect**
- **sp_helpsegment**
- **sp_helpserver**
- **sp_helpsort**
- **sp_helptext**
- **sp_helpthreshold**
- **sp_helpuser**
- **sp_lock**
- **sp_monitor**
- **sp_monitorconfig**
- **sp_procqmode**
- **sp_showcontrolinfo**
- **sp_showexeclass**

- **sp_helpjoins**
- **sp_helpkey**
- **sp_helplanguage**
- **sp_helplog**
- **sp_helpremotelogin**
- **sp_showplan**
- **sp_spaceused**
- **sp_who**
- **sp_help_resource_limit**

For complete information about the system procedures, see the *Adaptive Server Reference Manual*.

Keys in System Tables

Primary, foreign, and common keys for the system tables are defined in the *master* and *model* databases. You can get a report on defined keys by executing the system procedure **sp_helpkey.** For a report on columns in two system tables that are likely join candidates, execute **sp_helpjoins.**

Updating System Tables

The Adaptive Server system tables contain information that is critical to the operation of your databases. The data in these tables is inserted, updated, and deleted by Transact-SQL commands such as **create** and **drop** or by system procedures. Under ordinary circumstances, you do not need to perform direct data modifications to system tables.

Update system tables only when you are instructed to do so by Sybase Technical Support or by an instruction in the *Troubleshooting Guide* or in this manual.

When you update system tables, you must issue an **sp_configure** command that enables system table updates. While this command is in effect, any user with appropriate permission can modify a system table. Other requirements for direct changes to system tables are:

- Modify system tables only inside a transaction. Issue a **begin transaction** command before you issue the data modification command.
- Check to see that only the rows you wanted changed were affected by the command and that the data was changed correctly.

- If the command was incorrect, issue a **rollback transaction** command. If the command was correct, issue a **commit transaction** command.

◆ ***WARNING!***

Some system tables should not be altered by any user under any circumstances. Some system tables are built dynamically by system processes, contain encoded information, or display only a portion of their data when queried. Imprudent, ad hoc updates to certain system tables can make Adaptive Server unable to run, make database objects inaccessible, scramble permissions on objects, or terminate a user session.
Moreover, you should never attempt to alter the definition of the system tables in any way. For example, do not alter system tables to include constraints. Triggers, defaults, and rules are not allowed in system tables. If you try to create a trigger or bind a rule or default to a system table, you will get an error message.

System Procedures

The names of all system procedures begin with "sp_". They are located in the *sybsystemprocs* database, but many of them can be run from any database.

If a system procedure is executed in a database other than *sybsystemprocs*, it operates on the system tables in the database from which it was executed. For example, if the Database Owner of *pubs2* runs **sp_adduser** from *pubs2*, the new user is added to *pubs2..sysusers*. However, this does not apply to system procedures that update only tables in the *master* database.

Permissions on system procedures are discussed in the *Security Administration Guide* and the *Adaptive Server Reference Manual*.

Using System Procedures

A **parameter** is an argument to a stored or system procedure. If a parameter value for a system procedure contains reserved words, punctuation, or embedded blanks, it must be enclosed in single or double quotes. If the parameter is an object name, and the object name is qualified by a database name or owner name, the entire name must be enclosed in single or double quotes.

System procedures can be invoked by sessions using either chained or unchained transaction mode. However, the system procedures that modify data in system tables in the *master* database cannot be executed from within a transaction, since this could compromise recovery. The system procedures that create temporary work tables cannot be run from transactions.

If no transaction is active when you execute a system procedure, Adaptive Server turns off chained mode and sets **transaction isolation level 1** for the duration of the procedure. Before returning, the session's chained mode and isolation level are reset to their original settings. For more information about transaction modes and isolation levels, see the *Adaptive Server Reference Manual*.

All system procedures report a return status. For example:

```
return status = 0
```

means that the procedure executed successfully.

System Procedure Tables

The system procedures use several **system procedure tables** in the *master* and *sybsystemdb* databases to convert internal system values (for example, status bits) into human-readable format. One of these tables, *spt_values*, is used by a variety of system procedures, including:

- **sp_configure**
- **sp_dboption**
- **sp_depends**
- **sp_help**
- **sp_helpdb**
- **sp_helpdevice**
- **sp_helpindex**
- **sp_helpkey**
- **sp_helprotect**
- **sp_lock**

The *spt_values* table can be updated only by an upgrade; it cannot be modified otherwise. To see how it is used, execute **sp_helptext** and look at the text for one of the system procedures that references it.

The other system procedure tables are *spt_monitor*, *spt_committab*, and tables needed by the catalog stored procedures. (The *spt_committab* table is located in the *sybsystemdb* database.)

In addition, several of the system procedures create and then drop temporary tables. For example, **sp_helpdb** creates *#spdbdesc*, **sp_helpdevice** creates *#spdevtab*, and **sp_helpindex** creates *#spindtab*.

Creating System Procedures

Many of the system procedures are explained in this manual, in the sections where they are relevant. For complete information about system procedures, see the *Adaptive Server Reference Manual*.

System Administrators can write system procedures that can be executed from any database. Simply create a stored procedure in *sybsystemprocs* and give it a name that begins with "sp_". The *uid* of the stored procedure must be 1, the *uid* of the Database Owner.

Most of the system procedures that you create query the system tables. You can also create stored procedures that modify the system tables, although this is not recommended.

To create a stored procedure that modifies system tables, a System Security Officer must first turn on the **allowiupdates to system tables** configuration parameter. Any stored procedure created while this parameter is set to "on" will **always** be able to update system tables, even when **allow updates to system tables** is set to "off." To create a stored procedure that updates the system tables:

1. Use **sp_configure** to set **allow updates to system tables** to "on."
2. Create the stored procedure with the **create procedure** command.
3. Use **sp_configure** to set **allow updates to system tables** to "off."

◆ ***WARNING!***

Use extreme caution when you modify system tables. Always test the procedures that modify system tables in development or test databases, not in your production database.

System Extended Stored Procedures

An extended stored procedure (ESP) provides a way to call external language functions from within Adaptive Server. Adaptive Server provides a set of ESPs; users can also create their own. The names of all system extended stored procedures begin with "xp_". Like the system procedures, they are located in the *sybsystemprocs* database.

One very useful system ESP is **xp_cmdshell**, which executes an operating system command on the system that is running Adaptive Server.

A user invokes a system ESP just like a system procedure. The difference is that a system ESP executes procedural language code rather than Transact-SQL statements. All ESPs are implemented by an Open Server application called XP Server, which runs on the same machine as SQL Server. XP Server starts up automatically on the first ESP innovation.

For information about the system ESPs provided with Adaptive Server, see "System Extended Stored Procedures" in the *Adaptive Server Reference Manual.*

Creating System ESPs

To create a system ESP, create an ESP in the *sybsystemprocs* database using the **create procedure** command. (System procedures are automatically included in the *sybsystemprocs* database.) The name of the ESP, and its procedural language function, should begin with "xp_". The *uid* of the stored procedure must be 1, the *uid* of the Database Owner.

For general information about creating ESPs, see "Using Extended Stored Procedures" in the *Transact-SQL User's Guide.*

Logging Error Messages

Adaptive Server writes start-up information to a local error log file each time it boots. The name and location of the error log file is determined by a start-up parameter, **-e** on UNIX and PC platforms; **/errorfile** on OpenVMS. The installation program automatically sets the error log location when you configure a new Adaptive Server. See the the configuration documentation for your platform to learn the default location and file name of the error log.

Many error messages from Adaptive Server go to the user's terminal only. However, fatal error messages (severity levels 19 and above), kernel error messages, and informational messages from Adaptive Server are recorded in the error log file.

Adaptive Server keeps the error log file open until you stop the server process. If you need to reduce the size of the error log by deleting old messages, stop the Adaptive Server process before you do so.

Connecting to Adaptive Server

Adaptive Server can communicate with other Adaptive Servers, Open Server applications, and client software on the network. Clients can talk to one or more servers, and servers can communicate with other servers via remote procedure calls. In order for products to interact with one another, each needs to know where the others reside on the network. This network service information is stored in the interfaces file.

The Interfaces File

The interfaces file is usually named *interfaces*, *interfac*, or *sql.ini*, depending on the operating system.

The interfaces file is like an address book. It lists the name and address of every known server. When you use a client program to connect to a server, the program looks up the server name in the interfaces file and then connects to the server using the address, as shown in Figure 1-1.

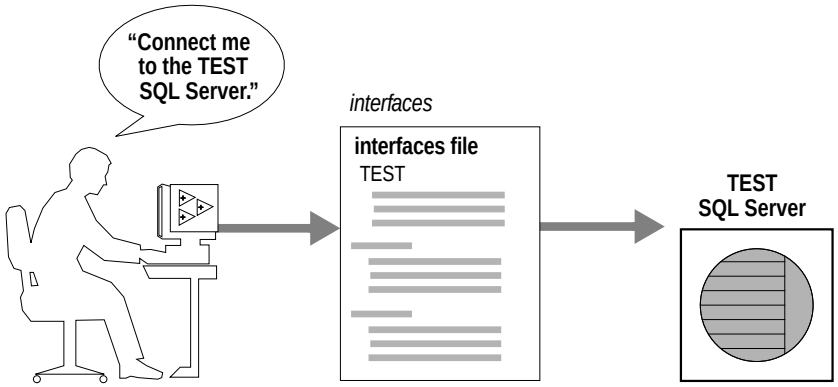


Figure 1-1: Connecting to Adaptive Server

The name, location, and contents of the interfaces file differ between operating systems. Also, the format of the Adaptive Server addresses in the interfaces file differs between network protocols.

When you install Adaptive Server, the installation program creates a simple interfaces file that you can use for local connections to Adaptive Server over one or more network protocols. As a System

Administrator, it is your responsibility to modify the interfaces file and distribute it to users so that they can connect to Adaptive Server over the network. See the configuration documentation for your platform for information about the interfaces file for your platform.

2 System Databases

This chapter describes the system databases that reside on all Adaptive Server systems. It also describes optional Sybase-supplied databases that you can install. Topics include:

- Overview of System Databases 2-1
- master Database 2-2
- model Database 2-4
- sybsystemprocs Database 2-5
- tempdb Database 2-6
- sybsecurity Database 2-7
- sybsystemdb Database 2-8
- pubs2 and pubs3 Sample Databases 2-8
- sybsyntax Database 2-9
- dbccdb Database 2-9

Overview of System Databases

When you install Adaptive Server, it includes these system databases:

- The *master* database
- The *model* database
- The system procedure database, *sybsystemprocs*
- The temporary database, *tempdb*

Optionally, you can install:

- The auditing database, *sybsecurity*
- The two-phase commit transaction database, *sybsystemdb*
- The sample databases, *pubs2* and *pubs3*
- The syntax database, *sybsyntax*
- The **dbcc** database, *dbccdb*

For information about installing the *master*, *model*, *sybsystemprocs*, and *tempdb* databases, see the installation documentation for your

platform. For information on installing *dbccdb*, see Chapter 18, "Checking Database Consistency."

The *master*, *model*, and temporary databases reside on the device named during installation, which is known as the master device. The *master* database is contained entirely on the master device and cannot be expanded onto any other device. All other databases and user objects should be created on other devices.

◆ ***WARNING!***

Do not store user databases on the master device. Storing user databases on the master device makes it difficult to recover the system databases if they become damaged. Also, you will not be able to recover user databases stored on the master device using the instructions in Chapter 22, "Restoring the System Databases."

The *sybsecurity* and *sybsystemdb* databases should be installed on their own devices and segment. For more information, see the installation documentation for your platform.

The *sybsystemprocs* database can be installed on a device of your choice. You may want to modify the installation scripts for *pubs2* and the *sybsyntax* to share the device you create for *sybsystemprocs*.

The *installpubs2* script does not specify a device in its create database statement, so it is created on the default device. At installation time, the master device is the default device. To change this, you can either edit the script or follow the instructions in this manual for adding more database devices and designating default devices. For more information, see Chapter 6, "Initializing Database Devices."

master Database

The *master* database controls the operation of Adaptive Server as a whole and stores information about all user databases and their associated database devices. Table 2-1 describes information the *master* database tracks:

Table 2-1: Information the master database tracks

Information	System Table
User accounts	*syslogins*
Remote user accounts	*sysremotelogins*

Table 2-1: Information the master database tracks (continued)

Information	System Table
Remote servers that this server can interact with	*sysservers*
Ongoing processes	*sysprocesses*
Configurable environment variables	*sysconfigures*
System error messages	*sysmessages*
Databases on Adaptive Server	*sysdatabases*
Storage space allocated to each database	*sysusages*
Tapes and disks mounted on the system	*sysdevices*
Active locks	*syslocks*
Character sets	*syscharsets*
Languages	*syslanguages*
Users who hold server-wide roles	*sysloginroles*
Server roles	*syssrvroles*
Adaptive Server engines that are online	*sysengines*

Because the *master* database stores information about user databases and devices, you must be in the *master* database in order to issue the **create database**, **alter database**, **disk init**, **disk refit**, **disk reinit**, and disk mirroring commands.

Controlling Object Creation in *master*

When you first install Adaptive Server, only a System Administrator can create objects in the *master* database, because the System Administrator implicitly becomes "dbo" of any database he or she uses. Any objects created on the *master* database should be used for the administration of the system as a whole. Permissions in *master* should remain set so that most users cannot create objects there.

◆ ***WARNING!***

Never place user objects in *master*. Storing user objects in *master* can cause the transaction log to fill quickly. If the transaction log runs out of space completely, you will not be able to use dump transaction commands to free space in *master*.

Another way to discourage users from creating objects in *master* is to change the default database for users (the database to which a user is connected when he or she logs in) with the system procedure **sp_modifylogin**. This procedure is discussed in Chapter 4, "Administering Roles," in the *Security Administration Guide*.

If you create your own system procedures, create them in the *sybsystemprocs* database rather than in *master*.

Backing Up *master* and Keeping Copies of System Tables

To be prepared for hardware or software failure on Adaptive Server, the two most important housekeeping tasks are:

- Performing frequent backups of the *master* database and all user databases. See "Keep Up-to-Date Backups of Master" on page 3-8 for more information. See also Chapter 22, "Restoring the System Databases," for an overview of the process for recovering the *master* database.
- Keeping a copy (preferably offline) of these system tables: *sysusages*, *sysdatabases*, *sysdevices*, *sysloginroles*, and *syslogins*. See "Keep Offline Copies of System Tables" on page 3-8 for more information. If you have copies of these scripts, and a hard disk crash or other disaster makes your database unusable, you can use the recovery procedures described in Chapter 22, "Restoring the System Databases." If you do not have current copies of your scripts, it will be much more difficult to recover Adaptive Server when the *master* database is damaged.

model Database

Adaptive Server includes the *model* database, which provides a template, or prototype, for new user databases. Each time a user enters the **create database** command, Adaptive Server makes a copy of the *model* database and extends the new database to the size specified by the **create database** command.

➤ ***Note***

A new database cannot be smaller than the *model* database.

The *model* database contains the required system tables for each user database. You can modify *model* to customize the structure of newly created databases—everything you do to *model* will be reflected in

each new database. Some of the changes that System Administrators commonly make to *model* are:

- Adding user-defined datatypes, rules, or defaults.
- Adding users who should have access to all databases on Adaptive Server.
- Granting default privileges, particularly for "guest" accounts.
- Setting database options such as **select into/bulkcopy/pllsort**. The settings will be reflected in all new databases. Their original value in *model* is **off**. For more information about the database options, see Chapter 16, "Setting Database Options."

Typically, most users do not have permission to modify the *model* database. There is not much point in granting read permission either, since Adaptive Server copies its entire contents into each new user database.

The size of *model* cannot be larger than the size of *tempdb*. Adaptive Server displays an error message if you try to increase the size of *model* without making *tempdb* at least as large.

➤ ***Note***

Keep a backup copy of the *model* database, and back up *model* with **dump database** each time you change it. In case of media failure, restore *model* as you would a user database.

sybsystemprocs Database

Sybase system procedures are stored in the database *sybsystemprocs*. When a user in any database executes any stored procedure that starts with the characters "sp_", Adaptive Server first looks for that procedure in the user's current database. If there is no procedure there with that name, Adaptive Server looks for it in *sybsystemprocs*. If there is no procedure in *sybsystemprocs* by that name, Adaptive Server looks for the procedure in *master*.

If the procedure modifies system tables (for example, **sp_adduser** modifies the *sysusers* table), the changes are made in the database from which the procedure was executed.

The **dbcc** stored procedures are not system procedures, although the names start with "sp_". They are stored in the *dbccdb* database. You must be in that database to use them and they modify only the tables

in that database. For more information, see Chapter 18, "Checking Database Consistency."

To change the default permissions on system procedures, you must modify those permissions in *sybsystemprocs*.

➤ ***Note***

If you make changes to *sybsystemprocs* or add your own stored procedures to the database, you should back up the database.

tempdb Database

Adaptive Server has a **temporary database**, *tempdb*. It provides a storage area for temporary tables and other temporary working storage needs (for example, intermediate results of **group by** and **order by**). The space in *tempdb* is shared among all users of all databases on the server.

The default size of *tempdb* is 2MB. Certain activities may make it necessary to increase the size of *tempdb*. The most common of these are:

- Large temporary tables.
- A lot of activity on temporary tables, which fills up the *tempdb* logs.
- Large sorts or many simultaneous sorts. Subqueries and aggregates with **group by** also cause some activity in *tempdb*.

You can increase the size of *tempdb* with the **alter database** command. *tempdb* is initially created on the master device. Space can be added from the master device or from any other database device.

Creating Temporary Tables

No special permissions are required to use *tempdb*, that is, to create temporary tables or to execute commands that may require storage space in the temporary database.

Create temporary tables either by preceding the table name in a **create table** statement with a pound sign (#) or by specifying the name prefix "tempdb..".

Temporary tables created with a pound sign are accessible only by the current Adaptive Server session: users on other sessions cannot

access them. These nonsharable, temporary tables are destroyed at the end of the current session. The first 13 bytes of the table's name, including the pound sign (#), must be unique. Adaptive Server assigns the names of such tables a 17-byte number suffix. (You can see the suffix when you query *tempdb..sysobjects*.)

Temporary tables created with the "tempdb.." prefix are stored in *tempdb* and can be shared among Adaptive Server sessions. Adaptive Server does not change the names of temporary tables created this way. The table exists either until you restart Adaptive Server or until its owner drops it using **drop table**.

System procedures (for example, **sp_help**) work on temporary tables, but only if you use them from *tempdb*.

If a stored procedure creates temporary tables, the tables are dropped when the procedure exits. Temporary tables can also be dropped explicitly before a session ends.

◆ ***WARNING!***

Do not create temporary tables with the "tempdb.." prefix from inside a stored procedure unless you intend to share those tables among other users and sessions.

Each time you restart Adaptive Server, it copies *model* to *tempdb*, which clears the database. Temporary tables are not recoverable.

sybsecurity Database

The *sybsecurity* database contains the audit system for Adaptive Server. It consists of:

- The system tables, *sysaudits_01*, *sysaudits_02*, ... *sysaudits_08*, which contain the audit trail
- The *sysauditoptions* table, which contains rows describing the global audit options
- All other default system tables that are derived from *model*

The audit system is discussed in more detail in Chapter 8, "Auditing," in the *Security Administration Guide*.

sybsystemdb Database

The *sybsystemdb* database stores information about two-phase commit transactions. The *spt_committab* table, which stores information about and tracks the completion status of each two-phase commit transaction, is stored in the *sybsystemdb* database.

Two-phase commit transactions and how to create the *sybsystemdb* database is discussed in detail in the configuration documentation for your platform.

pubs2 and *pubs3* Sample Databases

Installing the *pubs2* and *pubs3* sample databases is optional. These databases are provided as a learning tool for Adaptive Server. The *pubs2* sample database is used for most of the examples in the Adaptive Server documentation, except for examples, where noted, that use the *pubs3* database. For information about installing *pubs2* and *pubs3*, see the installation documentation for your platform. For information about the contents of these sample databases, see the *Transact-SQL User's Guide*.

Maintaining the Sample Databases

The sample databases contain a "guest" user mechanism that allows access to the database by any authorized Adaptive Server user. The "guest" user has been given a wide range of privileges in *pubs2* and *pubs3*, including permissions to select, insert, update, and delete user tables. For more information about the "guest" user mechanism and a list of the guest permissions in *pubs2* and *pubs3*, see Chapter 3, "Managing Adaptive Server Logins and Database Users," in the *Security Administration Guide*.

The *pubs2* and *pubs3* databases require at least 2MB each. If possible, you should give each new user a clean copy of *pubs2* and *pubs3* so that she or he is not confused by other users' changes. If you want to place *pubs2* or *pubs3* on a specific database device, edit the installation script before installing the database.

If space is a problem, you can instruct users to issue the **begin transaction** command before updating a sample database. After the user has finished updating one of the sample databases, he or she can issue the **rollback transaction** command to undo the changes.

pubs2 image Data

Adaptive Server includes a script for installing *image* data in the *pubs2* database. (*pubs3* does not use the image data.) The *image* data consists of six pictures, two each in PICT, TIF, and Sun raster file formats. Sybase does not provide any tools for displaying *image* data. You must use the appropriate screen graphics tools to display the images after you extract them from the database.

See the the installation documentation for your platform for information about installing the *image* data in *pubs2*.

sybsyntax Database

The syntax database, *sybsyntax*, contains syntax help for Transact-SQL commands, Sybase system procedures, Adaptive Server utilities, and Open Client™ DB-Library™ routines. Users can retrieve this information using the system procedure **sp_syntax.** For example, to learn the syntax of the Transact-SQL **select** command, type:

```
sp_syntax "select"
```

Adaptive Server includes two scripts for creating the *sybsyntax* database. The first script, usually named *ins_syn_sql*, installs syntax help for Transact-SQL commands, Sybase system procedures, and Adaptive Server utilities. The second script, usually named *ins_syn_dblib*, installs syntax help for Open Client DB-Library routines. You can install one or both of these scripts.

See the installation documentation for your platform for instructions on how to install *sybsyntax*. See the *Adaptive Server Reference Manual* for more information about **sp_syntax**.

dbccdb Database

dbcc checkstorage records configuration information for the **target database**, operation activity, and the results of the operation in the *dbccdb* database. Stored in the database are **dbcc** stored procedures for creating and maintaining *dbccdb* and for generating reports on the results of **dbcc checkstorage** operations. For more information, see Chapter 18, “Checking Database Consistency,” and Chapter 19, “dbccdb Tables.”

3 System Administration for Beginners

This chapter describes some of the basic tasks that System Administrators perform over the life of an Adaptive Server. Topics include:

- Using “Test” Servers 3-1
- Installing Sybase Products 3-3
- Allocating Physical Resources 3-4
- Backup and Recovery 3-7
- Ongoing Maintenance and Troubleshooting 3-11
- Keeping Records 3-11
- Getting More Help 3-14

This chapter:

- Introduces new System Administrators to important topics
- Helps System Administrators find information in the Sybase documentation

Experienced administrators may also find this chapter useful for organizing their ongoing maintenance activities.

You can find more information about the tasks presented in this chapter in other Adaptive Server manuals or in other Sybase documentation; references to individual books and chapters are included where applicable.

Using “Test” Servers

Whether you are a new or experienced System Administrator, it is always best to install and use a “test” and/or “development” Adaptive Server, with the intention of removing it before creating the “production” server. In general, using a test server makes it easier to plan and test different configurations and less stressful to recover from mistakes. It is much easier to learn how to install and administer new features when there is no risk of having to restart a production server or re-create a production database.

If you decide to use a test server, it is always best to do so from the point of installing or upgrading Adaptive Server through the process of configuring the server. It is in these steps that you make some of

the most important decisions about your final production system. The following sections describe the ways in which using a test server can help System Administrators.

Understanding New Procedures and Features

Using a test server allows you to practice basic administration procedures before performing them in a production environment. If you are a new Adaptive Server administrator, many of the procedures discussed in this book may be unfamiliar to you, and it may take several attempts to complete a task successfully. However, even experienced administrators will benefit from practicing techniques that are introduced by new features in Adaptive Server release 11.5.x Beta.

Planning Resources

While configuring a new server, administrators frequently discover that additional disks, memory, or hardware are needed to support their performance and throughput goals. Working with a test server helps you plan the final resource requirements for your system and helps you discover resource deficiencies that you might not have anticipated.

In particular, disk resources can have a dramatic effect on the final design of the production system. For example, you may decide that a particular database requires nonstop recovery in the event of a media failure. This would necessitate configuring one or more additional database devices to mirror the critical database. Discovering these resource requirements in a test server allows you to change the physical layout of databases and tables without affecting database users.

You can also use a test server as a “safe” environment for benchmarking both Adaptive Server and your applications using different hardware configurations. This allows you to determine the optimal setup for physical resources at both the Adaptive Server level and the operating system level before bringing the entire system online for general use.

Achieving Performance Goals

Although you can sometimes improve performance "on the fly" with production servers, most performance objectives can be met only by carefully planning a database's design and configuration. For example, you may discover that the insert and I/O performance of a particular table is a bottleneck. In this case, the best course of action may be to re-create the table on a dedicated segment and partition the table. Changes of this nature are disruptive to a production system, and even changing a configuration parameter can require you to restart Adaptive Server.

Installing Sybase Products

The responsibility for installing Adaptive Server and other Sybase products is sometimes placed with the System Administrator. If installation is one of your responsibilities, use the following pointers to help you in the process.

Check Product Compatibility

Before installing new products or upgrading existing products, always read the *Release Bulletin* included with the products to understand any compatibility issues that might affect your system. Compatibility problems can occur between hardware and software and between different release levels of the same software. Reading the *Release Bulletin* in advance can save the time and guesswork of troubleshooting known compatibility problems.

Also, refer to the lists of known problems that are installed with Adaptive Server. See "Learning About Known Problems" on page 4-25 for more information about these lists.

Install or Upgrade Adaptive Server

Read through the entire the installation documentation for your platform before you begin a new installation or upgrade. You need to plan parts of the installation and configure the operating system **before** installing Adaptive Server. It is also helpful to consult with the local administrator for your operating system to define operating system requirements for Adaptive Server. These requirements can include the configuration of memory, raw devices, asynchronous

I/O, and other features, depending on the platform you use. Many of these tasks cannot be accomplished once the installation has started.

If you are upgrading a server, back up all data (including the *master* database, user databases, triggers, and system procedures) offline before you begin. After upgrading, immediately create a separate, full backup of your data, especially if there are incompatibilities between older dump files and the newer versions.

Install Additional Third-Party Software

Adaptive Server generally includes support for the network protocol(s) that are common to your hardware platform. If your network supports additional protocols, install the required protocol support. As always, first read the *Release Bulletin* to make sure that the software is compatible with this Adaptive Server release.

Configure and Test Client Connections

A successful client connection depends on the coordination of Adaptive Server, the client software, and network products. If you are using one of the network protocols installed with Adaptive Server, see the configuration documentation for your platform for information about testing network connections. If you are using a different network protocol, follow the instructions that are included with the network product. You can also use "ping" utilities that are included with Sybase connectivity products to test client connections with Adaptive Server. For a general description of how clients connect to Adaptive Server, see "Connecting to Adaptive Server" on page 1-11. See also the configuration documentation for your platform for details about the name and contents of the interfaces file.

Allocating Physical Resources

Allocating physical resources is the process of giving Adaptive Server the memory, disk space, worker processes, and CPU power required to achieve your performance and recovery goals. When installing a new server, every System Administrator must make decisions about resource utilization. You will also need to reallocate Adaptive Server's resources if you then upgrade your hardware platform by adding new memory, disk controllers, or CPUs or when the design of your database system changes. Or, early benchmarking

of Adaptive Server and your applications can help you spot deficiencies in hardware resources that create performance bottlenecks.

See Chapter 5, "Overview of Disk Resource Issues," in this manual to understand the kinds of disk resources required by Adaptive Server. See also Chapter 8, "Configuring Memory," and Chapter 10, "Managing Multiprocessor Servers," for information about memory and CPU resources.

The following sections provide helpful pointers in determining physical resource requirements.

Dedicated vs. Shared Servers

The first step in the process of planning Adaptive Server resources is to understand the resources required by **other** applications running on the same machine. In most cases, System Administrators "dedicate" an entire machine for Adaptive Server use. This means that only the operating system and network software consume resources that otherwise might be reserved for Adaptive Server. On a "shared" system, other applications, such as Adaptive Server client programs or print servers, run on the same machine as Adaptive Server. It can be difficult to calculate the resources available to Adaptive Server on a shared system, because the types of programs and their pattern of use may change over time.

In either case, it is the System Administrator's responsibility to take into account the resources used by operating systems, client programs, windowing systems, and so forth when configuring resources for Adaptive Server. Configure Adaptive Server to use only the resources that are available to it. Otherwise, the server may perform poorly or fail to start. Chapter 8, "Configuring Memory," describes how to estimate the memory used by Adaptive Server. See also Chapter 5, "Overview of Disk Resource Issues," and Chapter 10, "Managing Multiprocessor Servers," to understand how the server utilizes disk and CPU resources.

Decision Support and OLTP Applications

Determine, in advance, the required mix between the amount of online transaction processing (OLTP) work and decision support work that Adaptive Server is expected to perform. Adaptive Server contains many features that optimize performance for OLTP, decision support, and mixed workload environments. However, you

must determine the requirements of your system's applications in order to make optimal use of these features.

For mixed workload systems, list the individual tables that you anticipate will be most heavily used for each type of application. Maintaining a list of critical tables can help you when configuring named caches or partitioning data to achieve maximum performance for applications.

Plan Resource Usage in Advance

It is extremely important to understand and plan resource usage in advance. In the case of disk resources, for example, once you initialize and allocate a device to Adaptive Server, that device cannot be used for any other purpose (even if Adaptive Server never fills the device with data). Likewise, Adaptive Server automatically reserves the memory for which it is configured, and this memory cannot be used by any other application.

The following suggestions can help you plan resource usage:

- For recovery purposes, it is **always** best to place a database's transaction log on a separate physical device from its data. See Chapter 15, "Creating and Managing User Databases."
- Consider mirroring devices that store mission-critical data. See Chapter 7, "Mirroring Database Devices." You may also consider using disk arrays and disk mirroring for Adaptive Server data if your operating system supports these features.
- If you are working with a test Adaptive Server, you can initialize database devices as operating system files, rather than raw devices, for convenience. This helps you determine accurate sizes for the raw devices that you will create for the production server. See the installation documentation for your platform for information about configuring operating system files vs. raw devices.
- Keep in mind that changing configuration options can affect the way Adaptive Server consumes physical resources. This is especially true of memory resources. See Chapter 11, "Setting Configuration Parameters," for details about how much memory is used by individual parameters.

Configure the Operating System

Many times, you need to configure physical resources at the operating system level before you can make them available to Adaptive Server. After determining what resources are available to Adaptive Server and what resources you require, configure these resources at the operating system level:

- Initialize raw devices to the sizes required by Adaptive Server. If you initialize a raw device for Adaptive Server, that device cannot be used for any other purpose (for example, it cannot be used to store operating system files). Ask your operating system administrator for assistance in initializing and configuring raw devices to the required sizes.
- Configure the number of network connections. The **number of user connections** parameter controls the maximum number of connections to Adaptive Server. However, you must make sure that the machine on which Adaptive Server runs can actually support that many connections. This is usually achieved by setting a variable in an operating system start-up file. See your operating system documentation.
- Additional configuration may be required for your operating system and the applications that you use. Read the installation documentation for your platform to understand the Adaptive Server operating system requirements. Also read your client software or consult with your engineers to understand the operating system requirements for your applications.

Backup and Recovery

Making regular backups of your databases is crucial to the integrity of your database system. Although Adaptive Server automatically recovers from system crashes (for example, power outages) or server crashes, only **you** can recover from data loss caused by media failure. Follow the basic guidelines below for backing up your system.

The following chapters describe how to develop and implement a backup and recovery plan:

- Chapter 20, “Developing a Backup and Recovery Plan”
- Chapter 21, “Backing Up and Restoring User Databases”
- Chapter 22, “Restoring the System Databases”
- Chapter 23, “Managing Free Space with Thresholds”

Keep Up-to-Date Backups of Master

Backing up the *master* database is the cornerstone of any backup and recovery plan. The *master* database contains details about the structure of your entire database system. Its keeps track of the Adaptive Server databases, devices, and device fragments that make up those databases. Because Adaptive Server needs this information during recovery, it is crucial to maintain an up-to-date backup copy of the *master* database at all times.

To ensure that your backup of *master* is always up to date, back up the database after each command that affects disks, storage, databases, or segments. This means you should back up *master* after performing any of the following procedures:

- Creating or deleting databases
- Initializing new database devices
- Adding new dump devices
- Using any device mirroring command
- Creating or dropping system stored procedures, if they are stored in *master*
- Creating, dropping, or modifying a segment
- Adding new Adaptive Server logins

To back up *master* to a tape device, start **isql** and enter the command:

```
dump database master to "tape_device"
```

where *tape_device* is the name of the tape device (for example, */dev/rmt0*).

Keep Offline Copies of System Tables

In addition to backing up *master* regularly, keep offline copies of the contents of the following system tables: *sysdatabases*, *sysdevices*, *sysusages*, *sysloginroles*, and *syslogins*. Do this by using the **bcp** or **wbcp** utility, described in the *Utility Programs* manual for your platform manual, and by storing a printed copy of the contents of each system table. You can create a printed copy by printing the output of the following queries:

```
select * from sysusages order by vstart
select * from sysdatabases
select * from sysdevices
```

```
select * from sysloginroles
select * from syslogins
```

If you have copies of these tables, and a hard disk crash or some other disaster makes your database unusable, you will be able to use the recovery procedures described in Chapter 22, "Restoring the System Databases."

You should also keep copes of all data definition language (DDL) scripts for user objects, as described under "Keeping Records" on page 3-11.

Automate Backup Procedures

Creating an automated backup procedure takes the guesswork out of performing backups and makes the procedure easier and quicker to perform. Automating backups can be as simple as using an operating system script or a utility (for example, the UNIX **cron** utility) to perform the necessary backup commands. Or you can automate the procedure further using thresholds, which are discussed in Chapter 23, "Managing Free Space with Thresholds."

Although the commands required to create an automated script vary, depending on the operating system you use, all scripts should accomplish the same basic steps:

1. Start **isql** and dump the transaction log to a holding area (for example, a temporary file).
2. Rename the dump file to a name that contains the dump date, time, and database name.
3. Make a note about the new backup in a history file.
4. Record any errors that occurred during the dump in a separate error file.
5. Automatically send mail to the System Administrator for any error conditions.

A script that completes the above tasks simplifies the process of making incremental backups. Periodically, you should also verify the database's consistency, using **dbcc**, dump the entire database to tape, and purge the incremental log files.

Verify Data Consistency Before Backing Up a Database

Having backups of a database sometimes is not enough—you must have consistent, **accurate** backups (especially for *master*). If you back up a database that contains internal errors, the database will have the same errors when you restore it.

Using the **dbcc** commands, you can check a database for errors before backing it up. **dbcc** commands verify that the storage of a database or database object "makes sense" to Adaptive Server. Always use **dbcc** commands to verify the integrity of a database before dumping it. If **dbcc** detects errors, correct them before dumping the database.

Over time, you can begin to think of running **dbcc** as insurance for your databases. If you discovered few or no errors while running **dbcc** in the past, you may decide that the risk of database corruption is small and that **dbcc** needs to be run only occasionally. Or, if the consequences of losing data are too high, you should continue to run **dbcc** commands each time you back up a database.

➤ ***Note***

For performance considerations, many sites choose to run **dbcc** checks outside of peak hours or on separate servers.

See Chapter 18, "Checking Database Consistency," for information about the **dbcc** command.

Monitor the Log Size

When the transaction log becomes nearly full, it may be impossible to use standard procedures to dump transactions and reclaim space. The System Administrator should monitor the log size and perform regular transaction log dumps (in addition to regular database dumps) to make sure this situation never occurs. Use the preferred method of setting up a threshold stored procedure that notifies you (or dumps the log) when the log reaches a certain capacity. See Chapter 23, "Managing Free Space with Thresholds," for information about using threshold procedures. It is also good to dump the transaction log just prior to doing a full database dump in order to shorten the time required to dump and load the database.

You can also monitor the space used in the log segment manually by using the **sp_helpsegment** stored procedure, as described under "Getting Information About Segments" on page 17-16.

Ongoing Maintenance and Troubleshooting

In addition to making regularly scheduled backups, the System Administrator performs the following maintenance activities throughout the life of a server.

Starting and Stopping Adaptive Server

Most system administrators automate the procedure for starting Adaptive Server to coincide with the start-up of the server machine. This can be accomplished by editing operating system start-up scripts or through other operating system procedures. See the configuration documentation for your platform to determine how to start and stop Adaptive Server.

Viewing and Pruning the Error Log

As a System Administrator, you should examine the contents of the error log on a regular basis to determine if any serious errors have occurred. You can also use operating system scripts to scan the error log for particular messages and to notify the System Administrator when specific errors occur. Checking the error log regularly helps you determine whether there are continuing problems of the same nature or whether a particular database device is going bad. See Chapter 4, “Diagnosing System Problems,” for more information about error messages and their severity.

The error log file can grow large over time, since Adaptive Server appends informational and status messages to it each time it starts up. You can periodically “prune” the log file by opening the file and deleting old records. Keeping the log file to a manageable size saves disk space and makes it easier to locate current errors.

Keeping Records

Keeping records about your Adaptive Server system is an important part of your job as a System Administrator. Accurate records of changes and problems that you have encountered can be a valuable reference when you are contacting Sybase Technical Support or recovering databases. More important, they can provide vital information for administrators who manage the Adaptive Server system in your absence. The following sections describe the kinds of records that are most valuable to maintain.

Contact Information

Maintain a list of contact information for yourself (the System Administrator) as well as the System Security Officer, Operator, and database owners on your system. Also, record secondary contacts for each role. Make this information available to all Adaptive Server users so that the appropriate contacts receive enhancement requests and problem reports.

Configuration Information

Ideally, you should create databases, create database objects, and configure Adaptive Server using script files that you later store in a safe place. Storing the script files after use makes it possible to re-create your entire system in the event of a disaster. It also allows you to re-create database systems quickly on new hardware platforms for evaluation purposes. If you use a third-party tool to perform system administration, remember to generate equivalent scripts after performing administration tasks.

Consider recording the following kinds of information:

- Commands used to create databases and database objects (DDL scripts)
- Commands that add new Adaptive Server logins and database users
- The current Adaptive Server configuration file, as described in “Using sp_configure with a Configuration File” on page 11-10
- The names, locations, and sizes of all files and raw devices initialized as database devices

It is also helpful to maintain a dated log of all changes to the Adaptive Server configuration. Mark each change with a brief description of when and why you made the change, as well a summary of the end result.

Maintenance Schedules

Keep a calendar of events for regularly scheduled maintenance activities. Such a calendar should list any of the procedures you perform at your site:

- Using **dbcc** to check database consistency
- Backing up user and system databases

- Monitoring the space left in transaction logs (if this is not done automatically)
- Dumping the transaction log
- Examining the error log contents for Adaptive Server, Backup Server™, and Adaptive Server Monitor™.
- Running the **update statistics** command (see"Index Statistics" on page 7-42 in the *Performance and Tuning Guide*)
- Examining auditing information, if the auditing option is installed
- Recompiling stored procedures
- Monitoring the resource utilization of the server machine

System Information

Record information about the hardware and operating system on which you run Adaptive Server. This can include:

- Copies of operating system configuration files or start-up files
- Copies of network configuration files (for example, the *hosts* and *services* files)
- Names and permissions for the Adaptive Server executable files and database devices
- Names and locations of the tape devices used for backups
- Copies of operating system scripts or programs for automated backups, starting Adaptive Server, or performing other administration activities

Disaster Recovery Plan

Consolidate the basic backup and recovery procedures, the hints provided in "Backup and Recovery" on page 3-7, and your personal experiences in recovering data into a concise list of recovery steps tailored to your system. This can be useful both to yourself and to other System Administrators who may need to recover a production system in the event of an emergency.

Getting More Help

The amount of new information that System Administrators must learn can sometimes seem overwhelming. However, there are several software tools that can help you in learning and facilitating basic administration tasks. These include Adaptive Server Monitor, used for monitoring server performance and other activities, and Sybase Central, which simplifies many administration tasks. Also available are many third-party software packages designed to help System Administrators manage daily maintenance activities.

4 Diagnosing System Problems

This chapter discusses diagnosing and fixing system problems. Topics include:

- How Adaptive Server Uses Error Messages to Respond to System Problems 4-1
- Adaptive Server Error Logging 4-4
- Backup Server Error Logging 4-13
- Killing Processes 4-14
- Configuring Adaptive Server to Save SQL Batch Text 4-17
- Shutting Down Servers 4-23
- Learning About Known Problems 4-25

How Adaptive Server Uses Error Messages to Respond to System Problems

When Adaptive Server encounters a problem—in an error message that describes whether the problem is caused by the user or the system—it displays information about the problem, how serious it is, and what you can do to fix it. The error message consists of:

- A **message number**, which uniquely identifies the error message
- A **severity level number** between 10 and 24, which indicates the type and severity of the problem
- An **error state number**, which allows unique identification of the line of Adaptive Server code at which the error was raised
- An **error message**, which tells you what the problem is, and may suggest how to fix it

For example, this is what happens if you make a typing error and try to access a table that does not exist:

```
select * from publisher

Msg 208, Level 16, State 1:
publisher not found. Specify owner.objectname or
use sp_help to check whether the object exists
(sp_help may produce lots of output).
```

In some cases, there can be more than one error message for a single query. If there is more than one error in a batch or query, Adaptive

Server usually reports only the first one. Subsequent errors are caught the next time you execute the batch or query.

The error messages are stored in *master..sysmessages*, which is updated with each new release of Adaptive Server. Here are the first few rows (from an Adaptive Server with us_english as the default language):

```
select error, severity, description
from sysmessages
where error >=101 and error <=106
and langid is null
```

```
error severity description
----- -------- ----------------------------------------------------
  101       15 Line %d: SQL syntax error.
  102       15 Incorrect syntax near '%.*s'.
  103       15 The %S_MSG that starts with '%.*s' is too long.
               Maximum length is %d.
  104       15 Order-by items must appear in the select-list if
               the statement contains set operators.
  105       15 Unclosed quote before the character string '%.*s'.
  106       16 Too many table names in the query. The maximum
               allowable is %d.

(6 rows affected)
```

You can generate your own list by querying *sysmessages*. Here is some additional information for writing your query:

- If your server supports more than one language, *sysmessages* stores each message in each language. The column *langid* is NULL for us_english and matches the *syslanguages.langid* for other languages installed on the server. For information about languages on your server, use **sp_helplanguage.**
- The *dlevel* column in *sysmessages* is currently unused.
- The *sqlstate* column stores the SQLSTATE value for error conditions and exceptions defined in ANSI SQL92.
- Message numbers 17000 and greater are system procedure error messages and message strings.

Error Messages and Message Numbers

The combination message number (*error*) and the language ID (*langid*) uniquely identifies each error message. Messages with the same message number but different language IDs are translations.

```
select error, description, langid
from sysmessages
where error = 101
error description                          langid
----- ------------------------------------ ------
  101 Line %d: SQL syntax error.              NULL
  101 Ligne %1!: erreur de syntaxe SQL.          1
  101 Zeile %1!: SQL Syntaxfehler.               2

(3 rows affected)
```

The error message text is a description of the problem. The descriptions often include a line number, a reference to a kind of database object (a table, column, stored procedure, and so forth), or the name of a particular database object.

In the *description* field of *sysmessages*, a percent sign (%) followed by a character or character string serves as a placeholder for these pieces of data, which Adaptive Server supplies when it encounters the problem and generates the error message. "%d" is a placeholder for a number; "%S_MSG" is a placeholder for a kind of database object; "%.*s"—all within quotes—is a placeholder for the name of a particular database object. Table 4-1 lists placeholders and what they represent.

For example, the *description* field for message number 103 is:

```
The %S_MSG that starts with '%.*s' is too long.
Maximum length is %d.
```

The actual error message as displayed to a user might be:

```
The column that starts with 'title' is too long.
Maximum length is 80.
```

For errors that you report, it is important to include the numbers, object types, and object names. (See "Reporting Errors" on page 4-12.)

Variables in Error Message Text

Table 4-1 explains the symbols that appear in the text provided with each error message explanation:

Table 4-1: Error text symbols key

Symbol	Stands For
%d, %D	Decimal number

Table 4-1: Error text symbols key (continued)

Symbol	Stands For
%x,%X,%.*x,%lx, %04x, %08lx	Hexadecimal number
%s	Null-terminated string
%.*s, %*s, %*.s	String, usually the name of a particular database object
%S_*type*	Adaptive Server-defined structure
%c	Single character
%f	Floating-point number
%ld	Long decimal
%lf	Double floating-point number

Adaptive Server Error Logging

Error messages from Adaptive Server are sent only to the user's screen.

The back trace from fatal error messages (severity levels 19 and higher) and error messages from the kernel are also sent to an error log file. The name of this file varies; see the configuration documentation for your platform or the *Utility Programs* manual for your platform.

➤ ***Note***

The error log file is owned by the user who installed Adaptive Server (or the person who started Adaptive Server after an error log was removed). Permissions or ownership problems with the error log at the operating system level can block successful start-up of Adaptive Server.

Adaptive Server creates an error log for you if one does not already exist. You specify the location of the error log at start-up with the *errorlogfile* parameter in the runserver file or at the command lline. The Sybase installation utility configures the runserver file with *$SYBASE/install* as the location of the error log if you do not choose an alternate location during installation. If you do not specify the location in the runserver file or at the command line, the location of the error log is the directory from which you start Adaptive Server. For more information about specifying the location of the error log,

see the the *Utility Programs* manual for your platform manual for your platform.

➤ ***Note***

Always start Adaptive Server from the same directory, or with the runserver file or the error log flag, so that you can locate your error log.

Each time you start a server, messages in the error log provide information on the success (or failure) of the start and the recovery of each database on the server. Subsequent fatal error messages and all kernel error messages are appended to the error log file. If you need to reduce the size of the error log by deleting old or unneeded messages, you must "prune" the log while Adaptive Server is shut down. For information about pruning the error log file, see "Viewing and Pruning the Error Log" on page 3-11.

Error Log Format

Entries in the error log include the following information:

- The engine involved for each log entry. The engine number is indicated by a 2-digit number. If only one engine is online, the display is "00."
- The family ID of the originating thread:
  - In serial processing, the display is "00000."
  - In **parallel processing**, the display is the server process ID number of the parent of the originating thread.
- The server process ID of the originating thread:
  - In serial processing, this is the server process ID number of the thread that generated the message. If the thread is a system task, then the display is "00000."
  - In parallel processing, this is the server process ID number of the originating thread.
- The date, displayed in the format *yyyy/mm/dd*, which allows you to sort error messages by date.
- The time, displayed in 24-hour format, which includes seconds and hundredths of a second.
- The word "server" or "kernel". This entry is for Sybase Technical Support use only.

- The error message itself.

Figure 4-1 shows two examples of a line from an error log:

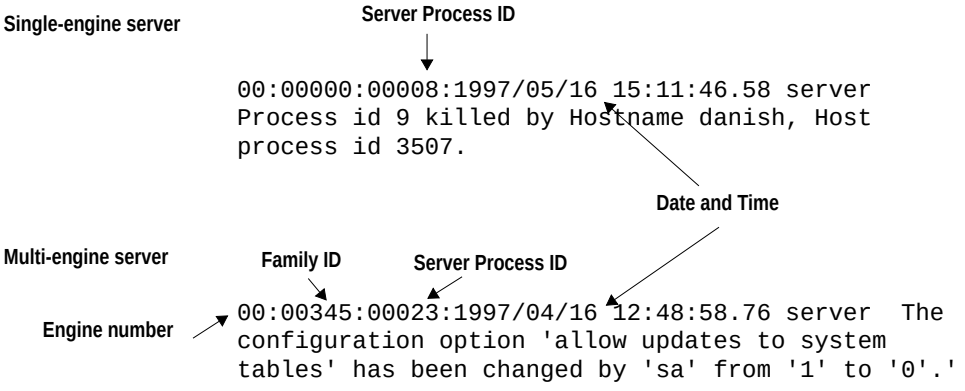


Figure 4-1: Error log format

Severity Levels

The severity level of a message indicates information about the type and severity of the problem that Adaptive Server has encountered. For maximum integrity, when Adaptive Server responds to error conditions, it displays messages from *sysmessages*, but takes action according to an internal table. A few corresponding messages differ in severity levels, so you may occasionally notice a difference in expected behavior if you are developing applications or procedures that refer to Adaptive Server messages and severity levels.

◆ ***WARNING!***

You can create your own error numbers and messages based on Adaptive Server error numbers (for example, by adding 20,000 to the Adaptive Server value). However, you cannot alter the Adaptive Server-supplied system messages in the *sysmessages* system table.

You can add user-defined error messages to *sysusermessages* with the system procedure **sp_addmessage**. See **sp_addmessage** in the *Adaptive Server Reference Manual.*

Users should be instructed to inform the System Administrator whenever problems that generate severity levels of 17 and higher occur. The System Administrator is responsible for resolving them and tracking their frequency. The System Administrator should monitor all problems that generate severity levels of 17–26.

If the problem has affected an entire database, the System Administrator may have to use the Database Consistency Checker (**dbcc**) to determine the extent of the damage. The **dbcc** may identify some objects that have to be removed. It can repair some damage, but the database may have to be reloaded.

For more information, refer to the following chapters:

- **dbcc** is discussed in Chapter 18, "Checking Database Consistency."
- Loading a user database is discussed in Chapter 21, "Backing Up and Restoring User Databases."
- Loading system databases is discussed in Chapter 22, "Restoring the System Databases."

The following sections discuss each severity level.

Levels 10–18

Error messages with severity levels 10–16 are generated by problems that are caused by user errors. These problems can always be corrected by the user. Severity levels 17 and 18 do not terminate the user's session.

Error messages with severity levels 17 and higher should be reported to the System Administrator or Database Owner.

Level 10: Status Information

Messages with severity level 10 are not errors at all. They provide additional information after certain commands have been executed and, typically, do not display the message number or severity level. For example, after a **create database** command has been run, Adaptive Server displays a message telling the user how much of the requested space has been allocated for the new database.

In **isql**, severity level 10 (status or informational) messages do not display a message number or severity level. In Open Client™ applications, Adaptive Server returns 0 for severity level 10.

Level 11: Specified Database Object Not Found

Messages with severity level 11 indicate that Adaptive Server cannot find an object that was referenced in the command.

This is often because the user has made a mistake in typing the name of a database object, because the user did not specify the object owner's name or because of confusion about which database is current. Check the spelling of the object names, use the owner names if the object is not owned by you or "dbo", and make sure you are in the correct database.

Level 12: Wrong Datatype Encountered

Messages with severity level 12 indicate a problem with datatypes. For example, the user may have tried to enter a value of the wrong datatype in a column or to compare columns of different (and incompatible) datatypes.

To correct comparison problems, use the **convert** function with **select**. For information on **convert**, see the *Adaptive Server Reference Manual* or the *Transact-SQL User's Guide*.

Level 13: User Transaction Syntax Error

Messages with severity level 13 indicate that something is wrong with the current user-defined transaction. For example, you may have issued a **commit transaction** command without having issued a **begin transaction** or you may have tried to roll back a transaction to a savepoint that has not been defined (sometimes there may be a typing or spelling mistake in the name of the savepoint).

Severity level 13 can also indicate a deadlock, in which case the deadlock victim's process is rolled back. The user must restart his or her command.

Level 14: Insufficient Permission to Execute Command

Messages with severity level 14 mean that you do not have the permission necessary to execute the command or access the database object. You can ask the owner of the database object, the owner of the database, or the System Administrator to grant you permission to use the command or object in question.

Level 15: Syntax Error in SQL Statement

Messages with severity level 15 indicate that the user has made a mistake in the syntax of the command. The text of these error messages includes the line numbers on which the mistake occurs and the specific word near which it occurs.

Level 16: Miscellaneous User Error

Most error messages with severity level 16 reflect that the user has made a nonfatal mistake that does not fall into any of the other categories. Severity level 16 and higher can also indicate software or hardware errors.

For example, the user may have tried to update a view in a way that violates the restrictions. Another error that falls into this category is unqualified column names in a command that includes more than one table with that column name. Adaptive Server has no way to determine which one the user intends. Check the command syntax and working database context.

Messages that ordinarily have severities greater than 16 will show severity 16 when they are raised by **dbcc checktable** or **dbcc checkalloc** so that checks can continue to the next object. When you are running the **dbcc** utility, check the *Error Messages* manual for information about error messages between 2500 and 2599 with a severity level of 16.

➤ ***Note***

The System Administrator should monitor occurrences of errors with severity levels 17–26. Levels 17 and 18 are usually not reported in the error log. Users should be instructed to notify the System Administrator when level 17 and 18 errors occur.

Level 17: Insufficient Resources

Error messages with severity level 17 mean that the command has caused Adaptive Server to run out of resources (usually space for the database on the disk) or to exceed some limit set by the System Administrator. You can continue with the work you are doing, although you may not be able to execute a particular command.

These system limits include the number of databases that can be open at the same time and the number of connections allowed to Adaptive Server. They are stored in system tables and can be

checked with the **sp_configure** command. See Chapter 11, "Setting Configuration Parameters," for more information on changing configuration variables.

The Database Owner can correct the level 17 error messages indicating that you have run out of space. Other level 17 error messages should be corrected by the System Administrator.

Level 18: Non-Fatal Internal Error Detected

Error messages with severity level 18 indicate some kind of internal software bug. However, the command runs to completion, and the connection to Adaptive Server is maintained. You can continue with the work you are doing, although you may not be able to execute a particular command. An example of a situation that generates severity level 18 is Adaptive Server detecting that a decision about the access path for a particular query has been made without a valid reason.

Since problems that generate such messages do not keep users from their work, users tend not to report them. Users should be instructed to inform the System Administrator every time an error message with this severity level (or higher) occurs so that the System Administrator can report them.

Severity Levels 19–24

Fatal problems generate error messages with severity levels 19 and higher. They break the user's connection to Adaptive Server. To continue working, the user must restart the client program.

When a fatal error occurs, the process (the program code that is running in order to accomplish the task that you specified in your command) is no longer running. The process freezes its state before it stops, recording information about what was happening. It is then killed and disappears.

When the user's connection is broken, he or she may or may not be able to reconnect and resume working. Some problems with severity levels in this range affect only one user and one process. Others affect all the processes in the database. In some cases, it will be necessary to restart Adaptive Server. These problems do not necessarily damage a database or its objects, but they can. They may also result from earlier damage to a database or its objects. Other problems are caused by hardware malfunctions.

A back trace of fatal error messages from the kernel is directed to the error log file, where the System Administrator can review it.

Level 19: Adaptive Server Fatal Error in Resource

Error messages with severity level 19 indicate that some non-configurable internal limit has been exceeded and that Adaptive Server cannot recover gracefully. You must reconnect to Adaptive Server. Users who receive this error message should report it to the System Administrator.

Level 20: Adaptive Server Fatal Error in Current Process

Error messages with severity level 20 mean that Adaptive Server has encountered a bug in a command. The problem has affected only the current process, and it is unlikely that the database itself has been damaged. Run **dbcc** diagnostics. You must reconnect to Adaptive Server. Users who receive this error message should report it to the System Administrator.

Level 21: Adaptive Server Fatal Error in Database Processes

Error messages with severity level 21 mean that Adaptive Server has encountered a bug that affects all the processes in the current database. However, it is unlikely that the database itself has been damaged. Restart Adaptive Server and run the **dbcc** diagnostics. You must reconnect to Adaptive Server. Users who receive this error message should report it to the System Administrator.

Level 22: Adaptive Server Fatal Error: Table Integrity Suspect

Error messages with severity level 22 mean that the table or index specified in the message was previously damaged by a software or hardware problem.

The first step is to restart Adaptive Server and run **dbcc** to determine whether other objects in the database are also damaged. Whatever the report from **dbcc** may be, it is possible that the problem is in the cache only and not on the disk itself. If so, restarting Adaptive Server will fix the problem.

If restarting does not help, then the problem is on the disk as well. Sometimes, the problem can be solved by dropping the object specified in the error message. For example, if the message tells you

that Adaptive Server has found a row with length 0 in a nonclustered index, the table owner can drop the index and re-create it.

You must reconnect to Adaptive Server. Users who receive this error message should report it to the System Administrator.

Level 23: Fatal Error: Database Integrity Suspect

Error messages with severity level 23 mean that the integrity of the entire database is suspect due to previous damage caused by a software or hardware problem. Restart Adaptive Server and run the **dbcc** diagnostics.

Even when a level 23 error indicates that the entire database is suspect, the damage may be confined to the cache, and the disk itself may be fine. If so, restarting Adaptive Server with **startserver** will fix the problem.

Level 24: Hardware Error or System Table Corruption

Error messages with severity level 24 reflect some kind of media failure or (in rare cases) the corruption of *sysusages*. The System Administrator may have to reload the database. It may be necessary to call your hardware vendor.

Level 26: Rule Error

Error messages with severity level 26 reflect that an internal locking or synchronization rule was broken. You must shut down and restart Adaptive Server.

Reporting Errors

When you report an error, be sure to include the following information:

- The message number, level number, and state number.
- Any numbers, database object types, or database object names that are included in the error message.
- The context in which the message was generated, that is, which command was running at the time. You can help by providing a hard copy of the backtrace from the error log.

Backup Server Error Logging

Like Adaptive Server, Backup Server creates an error log for you if one does not already exist.You specify the location of the error log at startup with the *error_log_file* parameter in the runserver file or at the command-line. The Sybase installation utility configures the runserver file with *$SYBASE/install* as the location of the error log if you do not choose an alternate location during installation. If you do not specify the location in the runserver file or at the command line, the location of the error log is the directory from which you start Backup Server. For more information about specifying the location of the error log, see the *Utility Programs* manual for your platform manual for your platform.

Backup Server error messages are in the form:

```
MMM DD YYY: Backup Server:N.N.N.N: Message Text
```

Backup Server message numbers consist of 4 integers separated by periods, in the form N.N.N.N. Messages in the form N.N.N are sent by Open Server™.

The four components of a Backup Server error message are *major.minor.severity.state*:

- The *major* component generally indicates the functional area of the Backup Server code where the error occurred:
  - 1 - System errors
  - 2 - Open Server event errors
  - 3 - Backup Server remote procedure call errors
  - 4 - I/O service layer errors
  - 5 - Network data transfer errors
  - 6 - Volume handling errors
  - 7 - Option parsing errors

  Major error categories 1–6 may result from Backup Server internal errors or a variety of system problems. Major errors in category 7 are almost always due to problems in the options you specified in your dump or load command.
- *minor* numbers are assigned in order within a major category.
- *severity* is one of the following:
  - 1 - Informational, no user action necessary.

  - 2, 3 - An unexpected condition, possibly fatal to the session, has occurred. The error may have occurred with any or all of usage, environment, or internal logic.
  - 4 - An unexpected condition, fatal to the execution of the Backup Server, has occurred. The Backup Server must exit immediately.
- *state* codes have a one-to-one mapping to instances of the error report within the code. If you need to contact Technical Support about Backup Server errors, the state code helps determine the exact cause of the error.

Killing Processes

A process is a task that is being carried out by Adaptive Server. Processes can be initiated by a user giving a command or by Adaptive Server itself. Each process is assigned a unique process identification number when it starts. These ID numbers, and other information about each process, are stored in *master..sysprocesses*. You can see most of the information by running the system procedure **sp_who**.

Running **sp_who** on a single-engine server shows the **sp_who** process "running" and all other processes "runnable" or in one of the sleep states. In multi-engine servers, there can be a "running" process for each engine.

The **kill** command gets rid of an ongoing process. The most frequent reason for killing a process is that it interferes with other users and the person responsible for running it is not available. The process may hold locks that block access to database objects, or there may be many sleeping processes occupying the available user connections. A System Administrator can kill processes that are:

- Waiting for an alarm, such as a **waitfor** command
- Waiting for network sends or receives
- Waiting for a lock
- Waiting for synchronization messages from another process in a family
- Most running or "runnable" processes

Adaptive Server allows you to kill processes only if it can cleanly roll back any uncompleted transactions and release all system resources

that are used by the process. For processes that are part of a family, all processes in the family must be killed simultaneously.

Table 4-2 shows the values that **sp_who** reports and when the **kill** command takes effect.

Table 4-2: Status values reported by sp_who

Status	Condition	Effects of *kill* Command
`recv sleep`	Waiting on a network read	Immediate.
`send sleep`	Waiting on a network send	Immediate.
`alarm sleep`	Waiting on an alarm such as `waitfor delay "10:00"`	Immediate.
`lock sleep`	Waiting on a lock acquisition	Immediate.
`sync sleep`	Waiting on a synchronization message from another process in the family.	Immediate. Other processes in the family must also be brought to a killable state.
`sleeping`	Waiting on a disk I/O, or some other resource. Probably indicates a process that is running, but doing extensive disk I/O	Killed when it "wakes up," usually immediate; a few sleeping processes do not wake up and require a Server reboot to clear.
`runnable`	In the queue of runnable processes	Immediate.
`running`	Actively running on one of the server engines	Immediate.
`infected`	Server has detected serious error condition; extremely rare	**kill** command not recommended. Server reboot probably required to clear process.
`background`	A process, such as a threshold procedure, run by Adaptive Server rather than by a user process	Immediate; use **kill** with extreme care. Recommend a careful check of *sysprocesses* before killing a background process.
`log suspend`	Processes suspended by reaching the last-chance threshold on the log	Immediate.

Only a System Administrator can issue the **kill** command: permission to use it cannot be transferred.

The syntax is:

```
kill spid
```

You can kill only one process at a time. A **kill** command is not reversible and cannot be included in a user-defined transaction. *spid* must be a numeric constant; you cannot use a variable. Here is some sample output from **sp_who**:

```
fid    spid   status       loginame     orignameo     hostname
blk dbname     cmd

 ------ ------ ------------ ------------ ------------ ----------
----- ---------- ----------------

     0      1 running      sa           sa           janetista
0

         master     SELECT

     0      2 sleeping     NULL         NULL
0


  fid  spid status   loginame hostname blk dbname cmd
  ---  ---- -------- -------- -------- --- ------ --------------
    0  1 recv sleep   bird    jazzy   0   master AWAITING COMMAND
    0  2  sleeping    NULL            0   master NETWORK HANDLER
    0  3  sleeping    NULL            0   master MIRROR HANDLER
    0  4  sleeping    NULL            0   master AUDIT PROCESS
    0  5  sleeping    NULL            0   master CHECKPOINT SLEEP
    0  6  recv sleep  rose    petal   0   master AWAITING COMMAND
    0  7  running     sa      helos   0   master SELECT
    0  8  send sleep  daisy   chain   0   pubs2  SELECT
    0  9  alarm sleep lily    pond    0   master WAITFOR
    0 10  lock sleep  viola   cello   7   pubs3  SELECT

(10 rows affected)
```

In the example above, processes 2–5 cannot be killed: they are system processes. The login name NULL and the lack of a host name identify them as system processes. You will always see NETWORK HANDLER, MIRROR HANDLER and CHECKPOINT SLEEP (or, rarely, CHECKPOINT). AUDIT PROCESS becomes activated if you enable auditing.

Processes 1, 6, 8, 9, and 10 can be killed, since they have the status values “recv sleep,” “send sleep,” “alarm sleep,” and “lock sleep.”

In **sp_who** output, you cannot tell whether a process whose status is “recv sleep” belongs to a user who is using Adaptive Server and may be pausing to examine the results of a command or whether the process indicates that a user has restarted a PC or other terminal, and

left a stranded process.You can learn more about a questionable process by querying the *sysprocesses* table for information. For example, this query shows the host process ID and client software used by process 8:

```
select hostprocess, program_name
    from sysprocesses
where spid = 8

 hostprocess program_name
 ----------- ----------------
 3993        isql
```

This query, plus the information about the user and host from the **sp_who** results, provides additional information for tracking down the process from the operating system level.

Using *sp_lock* to Examine Blocking Processes

In addition to **sp_who**, the system procedure **sp_lock** can help identify processes that are blocking other processes. If the *blk* column in the **sp_who** report indicates that another process has been blocked while waiting to acquire locks, **sp_lock** can display information about the blocking process. For example, process 10 in the **sp_who** output above is blocked by process 7. To see information about process 7, execute:

```
sp_lock 7
```

For more information about locking in Adaptive Server, see the *Performance and Tuning Guide*.

Configuring Adaptive Server to Save SQL Batch Text

Occasionally a query or procedure causes Adaptive Server Monitor to hang. Users with the System Administrator role can configure Adaptive Server to give Adaptive Server Monitor access to the text of the currently executing SQL batch. Viewing the SQL text of long-running batches helps you debug hung processes or fine-tune long statements that are heavy resource consumers.

Adaptive Server must be configured to collect the SQL batch text and write it to shared memory, where the text can be read by Adaptive Server Monitor Server (the server component of Adaptive Server Monitor). The client requests might come from Monitor Viewer, which is a plug-in to Sybase Central, or other Adaptive Server Monitor Server applications.

Configuring Adaptive Server to save SQL batch text also allows you to view the current query plan in showplan format (as you would see after setting **showplan** on). You can view the current query plan from within Adaptive Server; see "Viewing the Query Plan of a SQL Statement" on page 4-22. SQL batches are viewable only through Adaptive Server Monitor Server. See the Adaptive Server Monitor Server documentation for more information about displaying the batch text.

Because the query or procedure you are viewing may be nested within a batch of SQL text, the *sysprocesses* table now includes columns for the line number, statement number and *spid* of a hung statement to view its query plan.

By default, Adaptive Server is not configured to save SQL batch text, so you must configure Adaptive Server to allocate memory for this feature. Adaptive Server Monitor access to SQL has no effect on performance if you have not configured any memory to save SQL batches.

Allocating Memory for Batch Text

You can configure the amount of the SQL text batch you want to save. Initially, Adaptive Server is not configured to save any SQL text. Once text saving is enabled, Adaptive Server copies the subsequent SQL text batches to memory shared with SQL Server Monitor. Because each new batch clears the memory for the connection and overwrites the previous batch, you can view only currently executing SQL statements. The steps to configure Adaptive Server to start saving SQL text are as follows:

1. Configure the amount of SQL text retained in memory. (See "Configuring the Amount of SQL Text Retained in Memory" on page 4-19.)
2. Enable SQL Server to start saving SQL text. (See "Enabling Adaptive Server to Start Saving SQL Text" on page 4-20.)

➤ ***Note***

You must have System Administration privileges to configure and save SQL text batches

Configuring the Amount of SQL Text Retained in Memory

After installation, you must decide the maximum amount of SQL text that can be copied to shared memory. Consider the following to help you determine how much memory to allocate per user:

- SQL batches exceeding the allocated amount of memory are truncated without warning. This means that if you do not allocate enough memory for the batch statements, the text you are interested in viewing might be the section of the batch that is truncated, as illustrated in Figure 4-2.

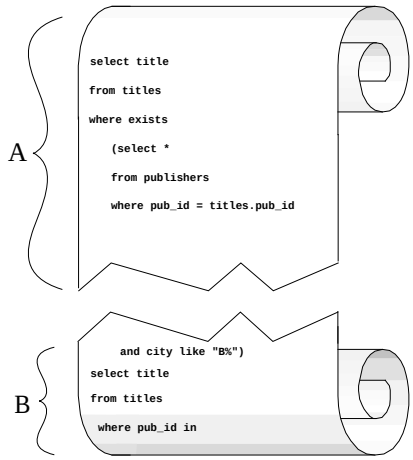


Figure 4-2: How SQL text is truncated if not enough memory is configured

  For example, if you configure Adaptive Server to save the amount of text designated by bracket A in the illustration, but the statement that is running occurs in the text designated by bracket B, Adaptive Server will not display the statement that is running.

- The more memory you allocate for SQL text from shared memory, the less chance the problem statement will be truncated from the batch copied to shared memory. However, the memory you allocate for SQL text reduces the amount of memory for data and procedure caches.

Sybase recommends an initial value of 1024 bytes per user connection. However, Adaptive Server immediately rejects very large values because they do not leave enough memory for data and procedure caches.

Use **sp_configure** with the **max SQL text monitored** configuration parameter to configure Adaptive Server to allocate shared memory. The syntax for the SQL text parameter is:

```
sp_configure "max SQL text monitored", bytes_per_connection
```

where *bytes_per_connection* (the maximum number of bytes saved for each client connection) is between 0 (the default) and 2,147,483,647 (the theoretical limit).

Since memory for SQL text is allocated by Adaptive Server at start-up, Adaptive Server must be restarted for this parameter to take affect.

The total memory allocated for the SQL text from shared memory is the product of *bytes_per_connection* multiplied by the currently configured number of user connections.

Enabling Adaptive Server to Start Saving SQL Text

Once you allocate shared memory for SQL text, Adaptive Server Monitor enables Adaptive Server to save a copy of each SQL batch whenever you enable an Adaptive Server Monitor event summary that includes SQL batches.

You may also have to reconfigure Adaptive Server Monitor's event buffer scan interval for SQL text. See the Adaptive Server Monitor documentation for more information.

SQL Commands Not Represented by Text

If you use Client-Library™ functions not represented by text (such as **ct_cursor** or **ct_dynamic**) to issue SQL commands, Client-Library encodes the information for efficiency, and Monitor Access to Executing SQL generally decodes and displays key command information. For example, if you open a cursor with **ct_cursor** and the command is running, the Adaptive Server Monitor event summary displays the cursor name and the cursor declare statement.

Table 4-3 lists a complete list of the Client-Library functions not represented by text:

Table 4-3: SQL commands not represented by text

Client-Library Routine	DB-Library Routine	Presentation Name	Presentation Data
ct_cursor	N / A	CLOSE_CURSOR	Cursor name, statement
ct_cursor	N / A	DECLARE_CURSOR	Cursor name, statement
ct_cursor	N / A	DELETE_AT_CURSOR	Cursor name, statement
ct_cursor	N / A	FETCH_CURSOR	Cursor name, statement
ct_fetch (when processing the results of ct_cursor)	N / A	FETCH_CURSOR	Cursor name, statement
ct_cursor CURSOR_ROWS, or ct_cancel when the connection has Client-Library cursors	N / A	CURSOR_INFO	Cursor name, statement
ct_cursor	N / A	OPEN_CURSOR	Cursor name, statement
ct_cursor	N / A	UPDATE_AT_CURSOR	Cursor name, statement
ct_command(CS_RPC_CMD) (default behavior)	**dbrpcinit** (only in release 10.0.1 or later)	DBLIB_RPC	RPC name
ct_dynamic	N / A	DYNAMIC_SQL	Dynamic statement name, statement
ct_command(CS_MSG_CMD	N / A	MESSAGE	None
ct_param(	**dbrpcparam**	PARAM_FORMAT	None
ct_param	**dbrpcparam**	PARAMS	None
ct_command(CS_RPC_CMD) (only when a TDS version earlier than 5.0 is used)	**dbrpcparam** (in DB-Library releases earlier than 10.0.1)	RPC	RPC name

For more information about SQL commands not represented by text, see your Open Client documentation.

Viewing the Query Plan of a SQL Statement

Use the **sp_showplan** system procedure and the *spid* of the user connection in question to retrieve the query plan for the statement currently running on this connection. You can also use **sp_showplan** to view the query plan for a previous statement in the same batch.

Following is the syntax for SQL Query Plan:

```
declare @batch int
declare @context int
declare @statement int
execute sp_showplan <spid_value>, @batch_id= @batch output,
@context_id= @context output, @stmt_num=@statement output
```

where *batch_id* is the unique number for a batch, *context_id* is a unique number for every procedure (or trigger) executed in the batch, and *stmt_num* is the number of the current statement within a batch. Adaptive Server uses the unique batch ID to synchronize the query plan with the batch text and other data retrieved by Adaptive Server Monitor.

➤ ***Note***

You must be a System Administrator to execute **sp_showplan**.

For example, if you wanted to see the query plan for the current statement for *spid* 99:

```
declare @batch int
declare @context int
declare @statement int
exec sp_showplan 99, @batch output, @context output, @statement output
```

You can run the query plan procedure independently of Adaptive Server Monitor, regardless of whether or not Adaptive Server has allocated shared memory for SQL text.

Viewing Previous Statements

To see the query plan for the previous statement in the same batch, issue **sp_showplan** with the same values as the original query, but subtract one from the statement number. Using this method, you can view all the statements in the statement batch back to query number one.

Viewing a Nested Procedure

Although **sp_showplan** allows you to view the query plan for the current statement, the actual statement that is running may exist within a procedure (or within a nested chain of procedures) called from the original SQL batch. The following columns were added to the *sysprocesses* table to access these nested statements:

Table 4-4: Columns added to sysprocesses

Column	Datatype	Specifies
id	Integer	The object ID of the running procedure (or 0 if no procedure is running).
stmtnum	Integer	The current statement number within the running procedure (or the SQL batch statement number if no procedure is running)
linenum	Integer	The line number of the current statement within the running stored procedure (or the line number of the current SQL batch statement if no procedure is running)

This information is saved in *sysprocesses*, regardless of whether SQL text is enabled or any memory is allocated for SQL text.

Following is an example of a SQL statement that displays the *id*, *stmtnum* and *linenum* columns of *sysprocesses*:

```
select id, stmtnum, linenum
from sysprocesses
where spid = spid_of_hung_session
```

➤ ***Note***

You do not need the **sa_role** to run this select statement.

Shutting Down Servers

A System Administrator can shut down Adaptive Server or Backup Server with the **shutdown** command. The syntax is:

```
shutdown [backup_server_name] [with {wait|nowait}]
```

The default for the **shutdown** command is **with wait.** That is, **shutdown** and **shutdown with wait** do exactly the same thing.

Shutting Down Adaptive Server

If you do not give a server name, **shutdown** shuts down the Adaptive Server you are using. When you issue a **shutdown** command, Adaptive Server:

1. Disables logins, except for System Administrators
2. Performs a checkpoint in each database, flushing pages that have changed from memory to disk
3. Waits for currently executing SQL statements or procedures to finish

In this way, **shutdown** minimizes the amount of work that automatic recovery must do when you restart Adaptive Server.

The **with no_wait** option shuts down Adaptive Server immediately. User processes are aborted, and recovery may take longer after a **shutdown with nowait.** You can help minimize recovery time by issuing a **checkpoint** command before you issue a **shutdown with nowait** command.

Shutting Down a Backup Server

To shut down a Backup Server, give the Backup Server's name:

```
shutdown SYB_BACKUP
```

The default is **with wait**, so any dumps or loads in progress will complete before the Backup Server process halts. Once you issue a **shutdown** command, no new dump or load sessions can be started on the Backup Server.

To see the names of the Backup Servers that are accessible from your Adaptive Server, execute **sp_helpserver**. Use the value in the *name* column in the **shutdown** command. You can shut down a Backup Server only if it is:

- Listed in *sysservers* on your Adaptive Server, and
- Listed in your local interfaces file.

Use **sp_addserver** to add a Backup Server to *sysservers*.

Checking for Active Dumps and Loads

To see the activity on your Backup Server before executing a **shutdown** command, run **sp_who** on the Backup Server:

```
SYB_BACKUP...sp_who
```

```
spid   status   loginame hostname   blk cmd
------ -------- -------- ---------- --- ----------------
     1 sleeping NULL     NULL       0   CONNECT HANDLER
     2 sleeping NULL     NULL       0   DEFERRED HANDLER
     3 runnable NULL     NULL       0   SCHEDULER
     4 runnable NULL     NULL       0   SITE HANDLER
     5 running  sa       heliotrope 0   NULL
```

Using *nowait* on a Backup Server

The **shutdown *backup_server* with nowait** command shuts down the Backup Server, regardless of current activity. Use it only in severe circumstances. It can leave your dumps or loads in incomplete or inconsistent states.

If you use **shutdown...with nowait** during a log or database dump, check for the message indicating that the dump completed. If you did not receive this message, or if you are not sure whether the dump completed, your next dump should be a **dump database**, not a transaction dump. This guarantees that you will not be relying on possibly inconsistent dumps.

If you use **shutdown with nowait** during a load of any kind, and you did not receive the message indicating that the load completed, you may not be able to issue further **load transaction** commands on the database. Be sure to run a full database consistency check (**dbcc**) on the database before you use it. You may have to reissue the full set of load commands, starting with **load database.**

Learning About Known Problems

The *Release Bulletin* is a valuable resource for learning about known problems or incompatibilities with Adaptive Server and Backup Server. Reading the *Release Bulletin* in advance can save you the time and guesswork of troubleshooting known problems.

The Adaptive Server installation program also installs files that list all System Problem Reports (SPRs) and Closed Problem Reports (CPRs) for Adaptive Server release 11.5.x Beta. Problem reports are organized by functional areas of the product. For example, a file named *cpr_bus* would contain a listing of closed (fixed) problem reports pertaining to the Backup Server, and the file *spr_bus* would contain a list of currently open problem reports for the Backup Server.

See the *Release Bulletin* to learn the location of CPR and SPR files.

Managing Physical Resources

5 Overview of Disk Resource Issues

This chapter discusses some basic issues that determine how you allocate and use disk resources with Adaptive Server. It covers the following topics:

- Device Allocation and Object Placement 5-1
- Commands for Managing Disk Resources 5-2
- Considerations in Storage Management Decisions 5-4
- Status and Defaults at Installation Time 5-5
- System Tables That Manage Storage 5-6

Adaptive Server can make reasonable default decisions about many aspects of storage management, such as where databases, tables, and indexes are placed and how much space is allocated for each one. However, the System Administrator has ultimate control over the allocation of disk resources to Adaptive Server and the physical placement of databases, tables, and indexes on those resources.

Responsibility for storage allocation and management is often centralized. However, in most installations, the System Administrator retains complete control over such matters.

Device Allocation and Object Placement

When configuring a new system, the System Administrator must consider several issues that have a direct impact on the number and size of disk resources required. These device allocation issues refer to commands and procedures that add disk resources to Adaptive Server. Device allocation topics are described in the chapters shown in Table 5-1.

Table 5-1: Device allocation topics

Task	Chapter
Initialize and allocate a default pool of database devices.	Chapter 6, “Initializing Database Devices”
Mirror database devices for recovery.	Chapter 7, “Mirroring Database Devices”

After the initial disk resources have been allocated to Adaptive Server, the System Administrator, Database Owner, and object owners should consider how to place databases and database objects on specific database devices. These object placement issues determine where database objects reside on your system and whether or not the objects share devices. Object placement tasks are discussed throughout this manual, including the chapters shown in Table 5-2.

Table 5-2: Object placement topics

Task	Chapter
Place databases on specific database devices.	Chapter 15, "Creating and Managing User Databases"
Place tables and indexes on specific database devices.	Chapter 17, "Creating and Using Segments"

The concept of allocating devices should not be considered separately from the concept of object placement. For example, if you decide that a particular table must reside on a dedicated pair of devices, you must first allocate those devices to Adaptive Server. The remaining sections in this chapter provide an overview that spans both device allocation and object placement issues, providing pointers to chapters where appropriate.

Commands for Managing Disk Resources

Table 5-3 illustrates the major commands a System Administrator uses to allocate disk resources to Adaptive Server and provides references to the chapters that discuss those commands.

Table 5-3: Commands for allocating disk resources

Command	Task	Chapter
disk init **name = "***dev_name***"** **physname = "***phys_name***"...**	Makes a physical device available to a particular Adaptive Server. Assigns a database device name (*dev_name*) that is used to identify the device in other Adaptive Server commands.	Chapter 6, "Initializing Database Devices"
sp_diskdefault "*dev_name***"...**	Adds *dev_name* to the general pool of default database space.	Chapter 6, "Initializing Database Devices"

Table 5-3: Commands for allocating disk resources (continued)

Command	Task	Chapter
disk mirror **name =** *"dev_name"* **mirror =** *"phys_name"*...	Mirrors a database device on a specific physical device.	Chapter 7, "Mirroring Database Devices"

The System Administrator, Database Owner, and object owners must decide how to place databases and database objects on specific devices. Table 5-4 illustrates the commands used in object placement. See also Chapter 17, "Controlling Physical Data Placement," in the *Performance and Tuning Guide* for information about how object placement affects performance.

Table 5-4: Commands for placing objects on disk resources

Command	Task	Chapter
create database...on *dev_name* or **alter database...on** *dev_name*	Makes database devices available to a particular Adaptive Server database. The **log on** clause to **create database** places the database's logs on a particular database device.	Chapter 15, "Creating and Managing User Databases"
create database... or **alter database...**	When used without the **on *dev_name*** clause, these commands allocate space on the default database devices.	Chapter 15, "Creating and Managing User Databases"
sp_addsegment *seg_name*, *dbname*, *devname* and **sp_extendsegment** *seg_name*, *dbname*, *devname*	Creates a segment, a named collection of space, from the devices available to a particular database.	Chapter 17, "Creating and Using Segments"
create table...on *seg_name* or **create index...on** *seg_name*	Creates database objects, placing them on a specific segment of the database's assigned disk space.	Chapter 17, "Creating and Using Segments"
create table... or **create index...**	When used without **on *seg_name***, tables and indexes occupy the general pool of space allocated to the database (the default devices).	Chapter 17, "Creating and Using Segments"

Considerations in Storage Management Decisions

The System Administrator of an Adaptive Server installation must make many decisions regarding the physical allocation of space to Adaptive Server databases. The major considerations in these choices are:

- **Recovery** – Disk mirroring and / or maintaining logs on a separate physical device provide two mechanisms for full recovery in the event of physical disk crashes.
- **Performance** – For certain tables or databases where speed of disk reads and writes is crucial, properly placing database objects to physical devices yields performance improvements. Disk mirroring slows the speed of disk writes.

Recovery

Recovery is the key motivation for using several disk devices. Nonstop recovery can be accomplished by mirroring database devices. Full recovery can also be ensured by storing a database's log on a separate physical device.

Keeping Logs on a Separate Device

Unless a database device is mirrored, full recovery in case of media failure requires that a database's transaction log be stored on a different device from the actual data (including indexes) of a database. You can use the **log on** clause of **create database** to guarantee that log records will be stored on a separate device. In the event of a hard disk crash, an up-to-date database can be re-created by loading a dump of the database and then applying the log records that were safely stored on another device. See Chapter 15, "Creating and Managing User Databases," for information about the **log on** clause of **create database**.

Mirroring

Nonstop recovery in the event of a hard disk crash is guaranteed through the mechanism of mirroring Adaptive Server devices to a separate physical disk. Mirroring the database devices containing the actual data and indexes (not just the device containing the transaction log) is required for recovery without downtime. Chapter

7, "Mirroring Database Devices," describes the process of mirroring devices.

Performance

System performance can be improved by placing logs and database objects on separate devices:

- Placing a table on one hard disk and nonclustered indexes on another ensures that physical reads and writes are faster, since the work is split between two disk drives.
- Splitting large tables across two disks can improve performance, particularly for multi-user applications.
- When log and data share devices, user log cache buffering of transaction log records is disabled.
- Partitioning provides multiple insertion points for a heap table, adds a degree of parallelism to systems configured to perform parallel query processing, and makes it possible to distribute a table's I/O over multiple database devices.

See Chapter 17, "Controlling Physical Data Placement," in the *Performance and Tuning Guide* for a detailed discussion of how object placement affects performance.

Status and Defaults at Installation Time

Instructions for installing Adaptive Server are provided in the installation documentation for your platform. The installation program and scripts initialize the master device and set up the *master*, *model*, *sybsystemprocs*, *sybsecurity*, and temporary databases for you.

When you first install Adaptive Server, the system databases, system defined segments, and database devices are organized as follows:

- The *master*, *model*, and *tempdb* databases are installed on the master device.
- The *sybsystemprocs* database is installed on a device that you choose at installation time.
- Three segments are created in each database: *system*, *default*, and *logsegment*.
- The master device is the default storage device for all user-created databases.

➤ ***Note***

After initializing new devices for default storage, remove the master device from the default storage area with **sp_diskdefault**. Do not store user databases and objects on the master device. See “Designating Default Devices” on page 6-7 for more information.

- If you have chosen to install the audit database, *sybsecurity*, it is located on its own device.

System Tables That Manage Storage

Two system tables in the *master* database and two more in each user database track the placement of databases, tables (including the transaction log table, *syslogs*), and indexes. The relationship between the tables is illustrated in Figure 5-1.

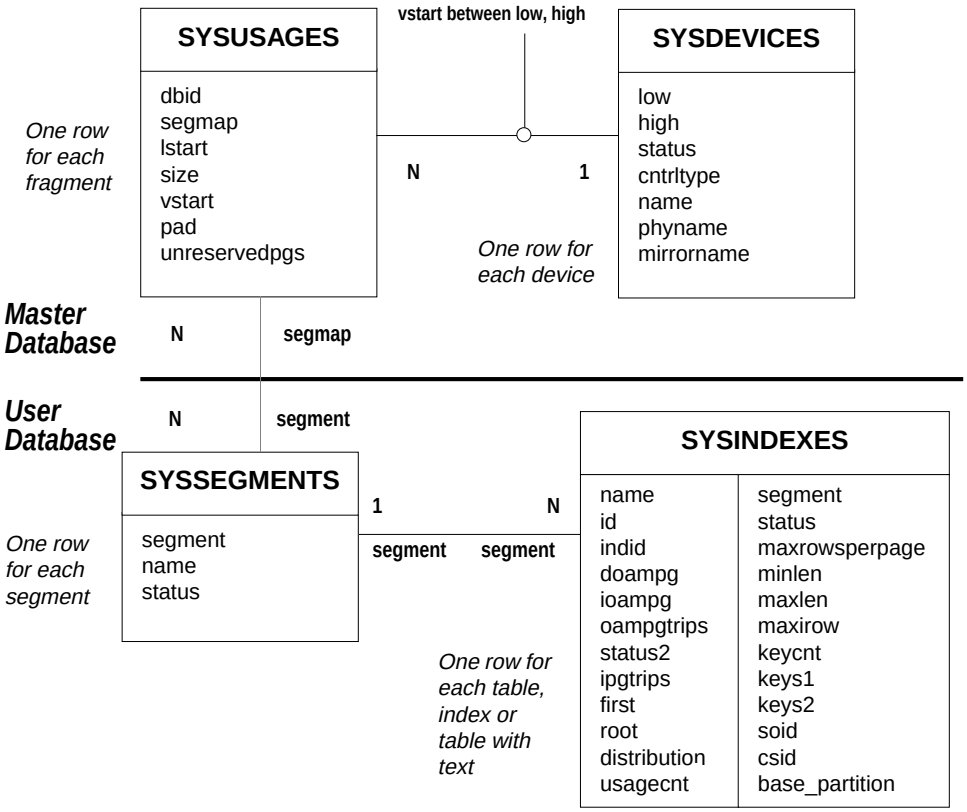


Figure 5-1: System tables that manage storage

The *sysdevices* Table

The *sysdevices* table in the *master* database contains one row for each **database device** and may contain a row for each dump device (tape, disk, or operating system file) available to Adaptive Server.

The **disk init** command adds entries for database devices to *master..sysdevices*. Dump devices, added with the system procedure **sp_addumpdevice**, are discussed in Chapter 20, "Developing a Backup and Recovery Plan."

sysdevices stores two names for each device:

- A **logical name** or **device name**, used in all subsequent storage-management commands, is stored in the *name* column of *sysdevices*. This is usually a user-friendly name, perhaps indicating the planned use for the device, for example "logdev" or "userdbdev".
- The **physical name** is the actual operating system name of the device. You use this name only in the **disk init** command; after that, all Adaptive Server data storage commands use the logical name.

You place a database or transaction log on one or more devices by specifying the logical name of the device in the **create database** or **alter database** statement. The **log on** clause to **create database** places a database's transaction log on a separate device to ensure full recoverability. The log device must also have an entry in *sysdevices* before you can use **log on**.

A database can reside on one or more devices, and a device can store one or more databases. See Chapter 15, "Creating and Managing User Databases," for information about creating databases on specific database devices.

The *sysusages* Table

The *sysusages* table in the *master* database keeps track of all of the space that you assign to all Adaptive Server databases.

create database and **alter database** allocate new space to the database by adding a row to *sysusages* for each database device or device fragment. When you allocate only a portion of the space on a device with **create** or **alter database**, that portion is called a **fragment**.

The system procedures **sp_addsegment**, **sp_dropsegment**, and **sp_extendsegment** change the *segmap* column in *sysusages* for the device that is mapped or unmapped to a segment. Chapter 17, "Creating and Using Segments," discusses these procedures in detail.

The *syssegments* Table

The *syssegments* table, one in each database, lists the segments in a database. A **segment** is a collection of the database devices and / or fragments available to a particular database. Tables and indexes can be assigned to a particular segment, and therefore to a particular physical device, or can span a set of physical devices.

create database makes default entries in *syssegments*. The system procedures **sp_addsegment** and **sp_dropsegment** add and remove entries from *syssegments*.

The *sysindexes* Table

The *sysindexes* table lists each table and index and the segment where each table, clustered index, nonclustered index, and chain of text pages is stored. It also lists other information such as the **max_rows_per_page** setting for the table or index.

The **create table**, **create index**, and **alter table** commands create new rows in *sysindexes*. Partitioning a table changes the function of *sysindexes* entries for the table, as described in Chapter 17, "Controlling Physical Data Placement," in the *Performance and Tuning Guide*.

6 Initializing Database Devices

This chapter explains how to initialize database devices and how to assign devices to the default pool of devices. Topics include:

- What Are Database Devices? 6-1
- Using the disk init Command 6-1
- disk init Syntax 6-2
- Getting Information About Devices 6-6
- Dropping Devices 6-7
- Designating Default Devices 6-7
- Placing Objects on Database Devices 6-9

What Are Database Devices?

A database device stores the objects that make up databases. The term **device** does not necessarily refer to a distinct physical device: It can refer to any piece of a disk (such as a disk partition) or a file in the file system that is used to store databases and their objects.

Each database device or file must be prepared and made known to Adaptive Server before it can be used for database storage. This process is called **initialization**.

Once a database device has been initialized, it can be:

- Allocated to the default pool of devices for the **create** and **alter database** commands
- Assigned to the pool of space available to a user database
- Assigned to a user database and used to store one or more database objects
- Assigned to store a database's transaction logs

Using the *disk init* Command

A System Administrator initializes new database devices with the **disk init** command. **disk init** does the following:

- Maps the specified physical disk device or operating system file to a **database device** name

- Lists the new device in *master..sysdevices*
- Prepares the device for database storage

➤ ***Note***

Before you run **disk init**, see the installation documentation for your platform for information about choosing a database device and preparing it for use with Adaptive Server. You may want to repartition the disks on your computer to provide maximum performance for your Sybase databases.

disk init divides the database devices into **allocation units** of 256 2K pages, a total of 1/2MB. In each 256-page allocation unit, the **disk init** command initializes the first page as the allocation page, which will contain information about the database (if any) that resides on the allocation unit.

◆ ***WARNING!***

After you run the disk init command, be sure to use dump database to dump the *master* database. This makes recovery easier and safer in case *master* is damaged. If you add a device and fail to back up *master*, you may be able to recover the changes with disk reinit. See Chapter 22, "Restoring the System Databases."

disk init Syntax

The syntax of the **disk init** command is:

```
disk init
   name = "device_name" ,
   physname = "physicalname" ,
   vdevno = virtual_device_number ,
   size = number_of_blocks
   [, vstart = virtual_address ,
        cntrltype = controller_number]
   [, contiguous] (OpenVMS only)
```

disk init Examples

On UNIX:

```
disk init
  name = "user_disk",
  physname = "/dev/rxy1a",
  vdevno = 2, size = 5120
```

On OpenVMS:

```
disk init
 name = "user_disk",
 physname = "disk$rose_1:[dbs]user.dbs",
 vdevno = 2, size = 5120,
 contiguous
```

On Windows NT:

```
disk init
  name = "user_disk",
  physname = "d:\devices\userdisk.dat",
  vdevno = 2, size = 5120
```

Specifying a Logical Device Name with *disk init*

The *device_name* must be a valid identifier. This name is used in the **create database** and **alter database** commands, and in the system procedures that manage segments. The logical device name is known only to Adaptive Server, not to the operating system on which the server runs.

Specifying a Physical Device Name with *disk init*

The *physicalname* of the database device gives the name of a raw disk partition (UNIX) or foreign device (OpenVMS) or the name of an operating system file. On PC platforms, you typically use operating system file names for *physicalname*.

Choosing a Device Number for *disk init*

vdevno is an identifying number for the database device. It must be unique among the devices used by Adaptive Server. Device number 0 represents the master device. Legal numbers are between 1 and 255, but the highest number must be one less than the number of database devices for which your system is configured. For example,

for a system with a default configuration of 10 devices, the legal device numbers are 1–9. To see the configuration value for your system, execute **sp_configure "number of devices"**, without giving a second parameter, and check the run value, the last column in the output:

```
sp_configure "number of devices"

Parameter name  Default  Memory Used  Config Value  Run Value
--------------- -------  -----------  ------------  ----------
number of devices    10           0            10          10
```

To see the numbers already in use for **vdevno**, look in the *device_number* column of the report from **sp_helpdevice**, or use the following query to list all the device numbers currently in use:

```
select distinct low/16777216
    from sysdevices
    order by low
```

Adaptive Server is originally configured for 10 devices. You may be limited to a smaller number of devices by operating system constraints. See the discussion of **sp_configure**, which is used to change configuration parameters, in Chapter 11, "Setting Configuration Parameters."

If you issue a **disk init** that does not succeed for any reason, and you need to reissue the **disk init** command, you must either use a different value for **vdevno** or restart Adaptive Server.

Specifying the Device Size with *disk init*

The **size** of the database device must be given in 2K blocks. There are 512 2K blocks in 1MB. The maximum size of a database device is system dependent.

➤ ***Note***

You cannot alter the size of a database device after running **disk init**. There is no equivalent to the **alterdb** command for a database device.

If you are planning to use the new device for the creation of a new database, the minimum **size** is the larger of the following:

- The size of *model*. When you install Adaptive Server, *model* uses 1024 2K blocks (2MB). Use **sp_helpdb model** to see the current size of *model*.

- The configuration parameter **default database size**. Use **sp_configure** and look at the run value for **default database size.**

If you are initializing a database device for a transaction log or for storing small tables or indexes on a segment, the **size** can be as small as 512 2K blocks (1MB).

If you are initializing a raw device (UNIX) or a foreign device (OpenVMS), determine the size of the device from your operating system, as described in the the installation documentation for your platform Use the total size available, up to the maximum of 2GB. Once you have initialized the disk for use by Adaptive Server, you cannot use any space on the disk for any other purpose. For this reason, you should never create a raw device or foreign device larger than 2GB if you plan to use it as a Adaptive Server database device.

disk init uses **size** to compute the value for the high virtual page number in *sysdevices.high*.

◆ ***WARNING!***

If the physical device does not contain the number of blocks specified by the size parameter, the disk init command fails. If you use the optional vstart parameter, the physical device must contain the sum of the blocks specified by both the vstart and size parameters, or the command fails.

If a **disk init** fails for any reason, you may need to restart Adaptive Server to free up the virtual device number. See the configuration documentation for your platform for information about starting and stopping Adaptive Server.

Optional Parameters for *disk init*

vstart is the starting virtual address, or the offset in 2K blocks, for Adaptive Server to begin using the database device. The default value (and usually the preferred value) of **vstart** is 0. If the specified device does not have the sum of **vstart** + **size** blocks available, the **disk init** command fails.

The optional **cntrltype** keyword specifies the disk controller. Its default value is 0. Reset it only if instructed to do so.

contiguous, an option for OpenVMS systems only, forces the database file to be created contiguously.

➤ ***Note***

To perform disk initialization, the user who started Adaptive Server must have the appropriate operating system permissions on the device that is being initialized.

Getting Information About Devices

The system procedure **sp_helpdevice** provides information about the devices in the *sysdevices* table.

When used without a device name, **sp_helpdevice** lists all the devices available on Adaptive Server. When used with a device name, it lists information about that device. Here, **sp_helpdevice** is used to report information about the master device:

```
sp_helpdevice master
device_name  physical_name  description
-----------  -------------- ----------------------------------------
master       d_master       special, default disk, physical disk, 20 MB
status      cntrltype    device_number     low      high
------      ----------   --------------    ------   -------
3           0            0                 0        9999
```

Each row in *master..sysdevices* describes:

- A dump device (tape, disk, or file) to be used for backing up databases, or
- A database device to be used for database storage.

The initial contents of *sysdevices* are operating system-dependent. Entries in *sysdevices* usually include:

- One for the master device
- One for the *sybsystemprocs* database, which you can use to store additional databases such as *pubs2* and *sybsyntax*, or for user databases and logs
- Two for tape dump devices

If you installed auditing, there will also be a separate device for *sybsecurity*.

The *low* and *high* fields represent the page numbers that have been assigned to the device. For dump devices, they represent the media capacity of the device.

The *status* field in *sysdevices* is a bitmap that indicates the type of device, whether a disk device will be used as a default storage device when users issue a **create** or **alter database** command without specifying a database device, and disk mirroring information. The status bits and their meanings are listed in the following table:

Table 6-1: Status bits in sysdevices

Bit	Meaning
1	Default disk (may be used by any **create** or **alter database** command that does not specify a location)
2	Physical disk
4	Logical disk (not used)
8	Skip header (used with tape dump devices)
16	Dump device
32	Serial writes
64	Device mirrored
128	Reads mirrored
256	Secondary mirror side only
512	Mirror enabled
2048	Used internally; set after **disk unmirror, side = retain**

For more information about dump devices and **sp_addumpdevice**, see Chapter 20, "Developing a Backup and Recovery Plan."

Dropping Devices

To drop database and dump devices, use the system procedure **sp_dropdevice**. The syntax is:

```
sp_dropdevice logicalname
```

You cannot drop a device that is in use by a database. You must drop the database first.

sp_dropdevice removes the device name from *sysdevices*. **sp_dropdevice** does not remove an operating system file that is being dropped as a database device: it only makes the file inaccessible to Adaptive Server. You must use operating system commands to delete a file after using **sp_dropdevice**.

Designating Default Devices

To create a pool of default database devices to be used by all Adaptive Server users for creating databases, use the **sp_diskdefault**

system procedure after the devices are initialized with **disk init**. **sp_diskdefault** marks these devices in *sysdevices* as default devices. Whenever users create databases (or alter databases) without specifying a database device, new disk space is allocated from the pool of default disk space.

The syntax for **sp_diskdefault** is:

```
sp_diskdefault logicalname, {defaulton | defaultoff}
```

You are most likely to use the **defaultoff** option to remove the master device from the pool of default space:

```
sp_diskdefault master, defaultoff
```

The following command makes *sprocdev*, the device that holds the *sybsystemprocs* database, a default device:

```
sp_diskdefault sprocdev, defaulton
```

Adaptive Server can have multiple default devices. They are used in the order in which they appear in the *sysdevices* table (that is, alphabetical order). When the first default device is filled, the second default device is used, and so on.

Choosing Default and Nondefault Devices

sp_diskdefault lets you plan space usage carefully for performance and recovery, while allowing users to create or alter databases occasionally.

Make sure these devices are **not** default devices:

- The master device (use **sp_diskdefault** to set **defaultoff** after adding user devices)
- The device for *sybsecurity*
- Any device intended solely for logs
- Devices where high-performance databases reside, perhaps using segments

You can use the device that holds *sybsystemprocs* for other user databases.

➤ ***Note***

If you are using disk mirroring or segments, you should exercise caution in deciding which devices you add to the default list with **sp_diskdefault**. In most cases, devices that are to be mirrored or databases that will contain objects placed on segments should allocate devices specifically, rather than being made part of default storage.

Placing Objects on Database Devices

After initializing a set of database devices, you may want to assign them to specific databases or database objects rather than adding them to the default pool of devices. For example, you may want to make sure a table never grows beyond the size of a particular device.

Refer to these chapters for information about placing databases and objects on devices:

- Chapter 15, "Creating and Managing User Databases," describes how to place user databases on specific database devices.
- Chapter 17, "Creating and Using Segments," explains how to use segments to place tables and indexes on specific devices.

You can also partition a table on a specific segment. For more information, see "Commands for Partitioning Tables" in Chapter 17, "Controlling Physical Data Placement," of the *Performance and Tuning Guide*.

7 Mirroring Database Devices

This chapter describes the process of creating and administering disk mirrors. Topics include:

- What Is Disk Mirroring? 7-1
- Deciding What to Mirror 7-1
- Conditions That Do Not Disable Mirroring 7-5
- Disk Mirroring Commands 7-6
- Disk Mirroring Tutorial 7-11

What Is Disk Mirroring?

Disk mirroring can provide nonstop recovery in the event of media failure. The **disk mirror** command causes a Adaptive Server database device to be duplicated, that is, all writes to the device are copied to a separate physical device. If one of the devices fails, the other contains an up-to-date copy of all transactions.

When a read or write to a mirrored device fails, Adaptive Server "unmirrors" the bad device and displays error messages. Adaptive Server continues to run unmirrored. To restart mirroring, the System Administrator must issue the **disk remirror** command.

Deciding What to Mirror

When deciding to mirror a device, you must weigh such factors as the costs of system downtime, possible reduction in performance, and the cost of storage media. Reviewing these issues will help you decide what to mirror—just the transaction logs, all devices on a server, or selected devices.

➤ ***Note***

You cannot mirror a dump device.

You should mirror all default database devices so that you are still protected if a **create** or **alter database** command affects a database device in the default list.

In addition to mirroring user database devices, you should always put their transaction logs on a separate database device. You can also mirror the database device used for transaction logs, for even greater protection.

To put a database's transaction log (that is, the system table *syslogs*) on a different device than the one on which the rest of the database is stored, name the database device and the log device when you create the database. You can also alter database to add a second device and then run the system procedure **sp_logdevice**.

Following are three examples which involve different cost and performance trade-offs:

1. **Speed of recovery** – You can achieve nonstop recovery when the *master* and user databases (including logs) are mirrored and can recover without the need to reload transaction logs.
2. **Storage space –** Immediate recovery requires full redundancy (all databases and logs mirrored), which consumes disk space.
3. **Impact on performance** – Mirroring the user databases (as in Figure 7-2 and Figure 7-3) increases the time needed to write transactions to both disks.

Mirroring Using Minimal Physical Disk Space

Figure 7-1 illustrates the "minimum guaranteed configuration" for database recovery in case of hardware failure. The master device and a mirror of the user database transaction log are stored in separate partitions on one physical disk. The other disk stores the user database and its transaction log in two separate disk partitions.

If the disk with the user database fails, you can restore the user database on a new disk from your backups and the mirrored transaction log.

If the disk with the master device fails, you can restore the master device from a database dump of the *master* database and remirror the user database's transaction log.

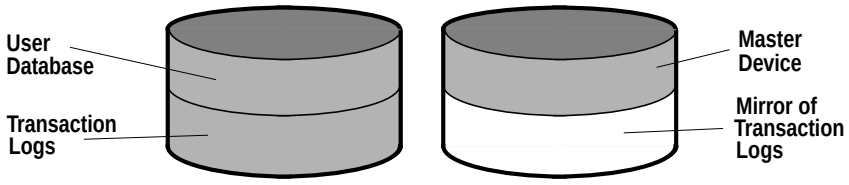


Figure 7-1: Disk mirroring using minimal physical disk space

This configuration minimizes the amount of disk storage required. It provides for full recovery, even if the disk storing the user database and transaction log is damaged, because the mirror of the transaction log ensures full recovery. However, this configuration does not provide nonstop recovery because the *master* and user databases are not being mirrored and must be recovered from backups.

Mirroring for Nonstop Recovery

Figure 7-2 represents another mirror configuration. In this case, the master device, user databases, and transaction log are all stored on different partitions of the same physical device and are all mirrored to a second physical device.

The configuration in Figure 7-2 provides nonstop recovery from hardware failure. Working copies of the *master* and user databases and log on the primary disk are all being mirrored, and failure of either disk will not interrupt Adaptive Server users.

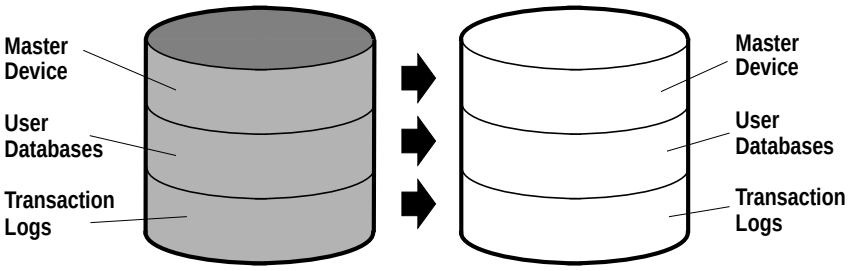


Figure 7-2: Disk mirroring for rapid recovery

With this configuration, all data is written twice, once to the primary disk and once to the mirror. Applications that involve many writes may be slower with disk mirroring than without mirroring.

Figure 7-3 illustrates another configuration with a high level of redundancy. In this configuration, all three database devices are mirrored, but the configuration uses four disks instead of two. This configuration speeds performance during write transactions because the database transaction log is stored on a different device from the user databases, and the system can access both with less disk head travel.

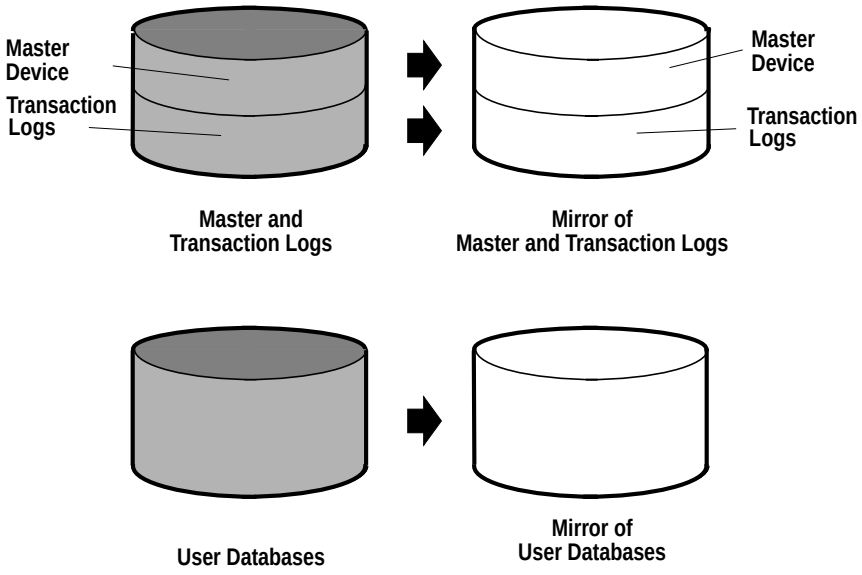


Figure 7-3: Disk mirroring: keeping transaction logs on a separate disk

Conditions That Do Not Disable Mirroring

Adaptive Server disables a mirror only when it encounters an I/O error on a mirrored device. For example, if Adaptive Server tries to write to a bad block on the disk, the resulting error disables mirroring for the device. However, processing continues without interruption on the unaffected mirror.

The following conditions **do not** disable a mirror:

- An unused block on a device is bad. Adaptive Server does not detect an I/O error and disables mirroring until it accesses the bad block.
- Data on a device is overwritten. This might happen if a mirrored device is mounted as a UNIX file system, and UNIX overwrites the Adaptive Server data. This causes database corruption, but mirroring is not disabled, since Adaptive Server would not encounter an I/O error.
- Incorrect data is written to both the primary and secondary devices.

- The file permissions on an active device are changed. Some System Administrators may try to test disk mirroring by changing permissions on one device, hoping to trigger I/O failure and unmirror the other device. But the UNIX operating system does not check permissions on a device after opening it, so the I/O failure does not occur until the next time the device is started.

Disk mirroring is not designed to detect or prevent database corruption. Some of the scenarios described can cause corruption, so you should regularly run consistency checks such as **dbcc checkalloc** and **dbcc checkdb** on all databases. See Chapter 18, "Checking Database Consistency," for a discussion of these commands.

Disk Mirroring Commands

The **disk mirror**, **disk unmirror**, and **disk remirror** commands control disk mirroring. All the commands can be issued while the devices are in use, so you can start or stop database device mirroring while databases are being used.

➤ ***Note***

The **disk mirror**, **disk unmirror**, and **disk remirror** commands alter the *sysdevices* table in the *master* database. After issuing any of these commands, you should dump the *master* database to ensure recovery in case *master* is damaged.

Initializing Mirrors

The **disk mirror** command starts disk mirroring. **Do not** initialize the mirror device with **disk init**. A database device and its mirror constitute one logical device. The **disk mirror** command adds the mirror name to the *mirrorname* column in the *sysdevices* table.

➤ ***Note***

To retain use of asynchronous I/O, always mirror devices that are capable of asynchronous I/O to other devices capable of asynchronous I/O. In most cases, this means mirroring raw devices to raw devices and operating system files to operating system files.

If the operating system cannot perform asynchronous I/O on files, mirroring a raw device to a regular file produces an error message. Mirroring a regular file to a raw device will work, but will not use asynchronous I/O.

Here is the **disk mirror** syntax:

```
disk mirror
   name = "device_name" ,
   mirror = "physicalname"
   [ , writes = { serial | noserial }]
   [ , contiguous ]      (OpenVMS only)
```

The *device_name* is the name of the device that you want to mirror, as it is recorded in *sysdevices.name* (by **disk init**). Use the **mirror =** "*physicalname*" clause to specify the path to the mirror device, enclosed in single or double quotes. If the mirror device is a file, "*physicalname*" must unambiguously identify the path where Adaptive Server will create the file; it cannot specify the name of an existing file.

On systems that support asynchronous I/O, the **writes** option allows you to specify whether writes to the first device must finish before writes to the second device begin (**serial**) or whether both I/O requests are to be queued immediately, one to each side of the mirror (**noserial**). In either case, if a write cannot be completed, the I/O error causes the bad device to become unmirrored.

serial writes are the default. The writes to the devices take place consecutively, that is, the first one finishes before the second one starts. **serial** writes provide protection in the case of power failures: One write may be garbled, but both of them will not be. **serial** writes are generally slower than **noserial** writes.

OpenVMS users should see the *Adaptive Server Reference Manual* for an explanation of the **contiguous** option.

In the following example, *tranlog* is the logical device name for a raw device. The *tranlog* device was initialized with **disk init** and is being used as a transaction log device (as in **create database...log on** *tranlog*). The following command mirrors the transaction log device:

```
disk mirror
  name = "tranlog",
  mirror = "/dev/rxy1e"
```

Effects on System Tables

The database device that you want to mirror will already have been initialized with **disk init**. Adaptive Server stores the physical name of the mirror device in the *mirrorname* column of the *sysdevices* table and updates the *status* bits to reflect the configuration. Table 6-1 on page 6-7 describes the status bits.

Unmirroring a Device

Disk mirroring is automatically deactivated when one of the two physical devices fails. When a read or write to a mirrored device is unsuccessful, Adaptive Server causes the bad device to become unmirrored and prints error messages. Adaptive Server continues to run, unmirrored. You must remirror the disk to restart mirroring.

Use the **disk unmirror** command to stop the mirroring process when hardware maintenance is needed or when a hardware device needs to be changed:

```
disk unmirror
     name = "device_name"
     [, side = { "primary" | secondary }]
     [, mode = { retain | remove }]
```

The **side** option to the **disk unmirror** command allows you to specify which side of the mirror to disable. **primary** (in quotes) is the device listed in the *name* column of *sysdevices*; **secondary** (no quotes required) is the device listed in the *mirrorname* column of *sysdevices*. **secondary** is the default.

The **mode** option indicates whether the unmirroring process should be temporary (**retain**) or permanent (**remove**). **retain** is the default.

Temporarily Deactivating a Device

By default (**mode**=**retain**), Adaptive Server deactivates the specified device temporarily; you can reactivate it later. This is similar to what happens when a device fails and Adaptive Server activates its mirror:

- I/O is directed only at the remaining device of the mirrored pair.

- The *status* column of *sysdevices* is altered to indicate that the mirroring feature has been deactivated.
- The entries for primary *(phyname)* and secondary *(mirrorname*) disks are unchanged.

Permanently Disabling a Mirror

Use **mode**=**remove** to completely disable disk mirroring. This option eliminates all references in the system tables to a mirror device, but does **not** remove an operating system file that has been used as a mirror.

If you set **mode**=**remove**:

- The *status* column is altered to indicate that the mirroring feature is to be completely ignored.
- The *phyname* column is replaced by the name of the secondary device in the *mirrorname* column if the primary device is the one being deactivated.
- The *mirrorname* column is set to NULL.

Effects on System Tables

The **mode** option changes the *status* column in *sysdevices* to indicate that mirroring has been disabled (see Table 6-1 on page 6-7). Its effects on the *phyname* and *mirrorname* columns in *sysdevices* depend on the **side** argument also, as shown in Table 7-1.

Table 7-1: Effects of mode and side options to the disk mirror command

		side	
		primary	**secondary**
mode	**remove**	Name in *mirrorname* moved to *phyname* and *mirrorname* set to null; *status* changed	Name in *mirrorname* removed; *status* changed
	retain	Names unchanged; *status* changed to indicate which device is being deactivated	

This example suspends the operation of the primary device:

```
disk unmirror
  name = "tranlog",
  side = primary
```

Restarting Mirrors

Use **disk remirror** to restart a mirror process that has been suspended due to a device failure or with **disk unmirror**. The syntax is:

```
disk remirror
     name = "device_name"
```

This command copies the database device to its mirror.

waitfor mirrorexit

Since disk failure can impair system security, the **waitfor mirrorexit** command can be included in an application to perform specific tasks when a disk becomes unmirrored:

```
begin
   waitfor mirrorexit
    commands to be executed
end
```

The commands depend on your applications. You may want to add certain warnings in applications that perform updates or use **sp_dboption** to make certain databases read only if the disk becomes unmirrored.

➤ ***Note***

Adaptive Server knows that a device has become unmirrored only when it attempts I/O to the mirror device. On mirrored databases, this occurs at a checkpoint or when the Adaptive Server buffer must be written to disk. On mirrored logs, I/O occurs when a process writes to the log, including any committed transaction that performs data modification, a checkpoint, or a database dump.

waitfor mirrorexit and the error messages that are printed to the console and error log on mirror failure are activated only by these events.

Mirroring the Master Device

If you choose to mirror the device that contains the *master* database, in a UNIX or Open VMS environment, you need to edit the runserver file for your Adaptive Server so that the mirror device starts when the server boots.

On UNIX, add the **-r** flag and the name of the mirror device:

```
dataserver -d /dev/rsd1f -r /dev/rs0e -e/sybase/install/errorlog
```

On OpenVMS, add the mirror name:

```
dataserver /device=(DUA0:[dbdevices]master.dat, -
     DUB1:[dbmirrors]mirror.dat) -
     /errorfile=sybase_system:[sybase.install]errorlog
```

For information about mirroring the master device on Windows NT, see the *Utility Programs* manual for your platform manual.

Getting Information About Devices and Mirrors

For a report on all Adaptive Server devices on your system (user database devices and their mirrors, as well as dump devices), execute the system procedure **sp_helpdevice**.

Disk Mirroring Tutorial

The following steps illustrate the use of disk mirroring commands and their effect on selected columns of *master..sysdevices*:

Step 1

Initialize a new test device using the command:

```
disk init name = "test",
physname = "/usr/sybase/test.dat",
size=5120, vdevno=3
```

This inserts the following values into columns of *master..sysdevices*:

```
name  phyname               mirrorname   status
test  /usr/sybase/test.dat   NULL         2
```

Status 2 indicates that the device is a physical disk. Since the device mirrored bit (64) is off and the *mirrorname* column is null, this device is not mirrored.

Step 2

Mirror the test device using the command:

```
disk mirror name = "test",
mirror = "/usr/sybase/test.mir"
```

This changes the *master..sysdevices* columns to:

```
name  phyname               mirrorname            status
test  /usr/sybase/test.dat  /usr/sybase/test.mir  738
```

Status 738 indicates that mirroring is currently active (512) on this device. Reads are mirrored (128), and writes are mirrored (64) and serial (32). The device is a physical disk (2).

Step 3

Disable the mirror device (the secondary side), but retain that mirror:

```
disk unmirror name = "test",
side = secondary, mode = retain
```

```
name  phyname               mirrorname            status
test  /usr/sybase/test.dat  /usr/sybase/test.mir  2274
```

Status 2274 indicates that the mirror device has been retained (2048), but mirroring has been disabled (512 bit off), and only the primary device is used (256 bit off). Reads are mirrored (128), and writes are mirrored (64) and serial (32). The device is a physical disk (2).

Step 4

Remirror the test device:

```
disk remirror name = "test"
```

This resets the *master..sysdevices* columns to:

```
name  phyname               mirrorname            status
test  /usr/sybase/test.dat  /usr/sybase/test.mir  738
```

Status 738 indicates that mirroring is currently active (512) on this device. Reads are mirrored (128), and writes are mirrored (64) and serial (32). The device is a physical disk (2).

Step 5

Disable the test device (the primary side), but retain that mirror:

```
disk unmirror name = "test",
side = "primary", mode = retain
```

This changes the *master..sysdevices* columns to:

```
name  phyname               mirrorname           status
test  /usr/sybase/test.dat  /usr/sybase/test.mir 482
```

Status 482 indicates that mirroring has been disabled (512 bit off) and that only the secondary device is used (256). Reads are mirrored (128), and writes are mirrored (64) and serial (32). The device is a physical disk (2).

Step 6

Remirror the test device:

```
disk remirror name = "test"
```

This resets the *master..sysdevices* columns to:

```
name  phyname               mirrorname           status
test  /usr/sybase/test.dat  /usr/sybase/test.mir  738
```

Status 738 indicates that mirroring is currently active (512) on this device. Reads are mirrored (128), and writes are mirrored (64) and serial (32). The device is a physical disk (2).

Step 7

Disable the test device (the primary side), and remove that mirror:

```
disk unmirror name = "test", side = "primary",
mode = remove
```

This changes the *master..sysdevices* columns to:

```
name  phyname               mirrorname           status
test  /usr/sybase/test.mir  NULL                 2
```

Status 2 indicates that the device is a physical device. Since the *mirrorname* column is null, mirroring is not enabled on this device.

Step 8

Remove the test device to complete the tutorial:

```
sp_dropdevice test
```

This removes all entries for the test device from *master..sysdevices*.

8 Configuring Memory

This chapter describes how Adaptive Server uses memory and explains how to maximize the memory available to Adaptive Server on your system. Topics include:

- Maximizing Adaptive Server Memory 8-1
- How Adaptive Server Uses Memory 8-2
- System Procedures for Configuring Memory 8-4
- Major Uses of Adaptive Server Memory 8-8
- Other Parameters That Use Memory 8-13

Maximizing Adaptive Server Memory

The more memory that is available, the more resources Adaptive Server has for internal buffers and caches. Having enough memory available for caches reduces the number of times Adaptive Server has to read data or procedure plans from disk.

There is no performance penalty for configuring Adaptive Server to use the maximum amount of memory available on your computer. However, be sure to assess the other memory needs on your system first, and then configure the Adaptive Server to use only the remaining memory that is still available. Adaptive Server may not be able to start if it cannot acquire the memory for which it is configured.

To determine the maximum amount of memory available on your system for Adaptive Server:

1. Determine the total amount of physical memory on your computer system.
2. Subtract the memory required for the operating system from the total physical memory.
3. Subtract the memory required for Backup Server, Monitor Server, or other Adaptive Server-related software that must run on the same machine.
4. If the machine is not dedicated to Adaptive Server, also subtract the memory requirements for other system uses.

   For example, subtract the memory that will be used by any client applications that will run on the Adaptive Server machine.

Windowing systems, such as X Windows, require a lot of memory and can interfere with Adaptive Server performance when used on the same machine as Adaptive Server.

5. Subtract any memory that you want to allocate for the **additional network memory** configuration parameter. This is explained in "additional network memory" on page 11-89.

The memory left over after subtracting requirements for the operating system, other applications, and **additional network memory** is the total memory available for Adaptive Server. Configure Adaptive Server to use this remaining memory by setting the **total memory** parameter to that value. See "total memory" on page 11-92 for details on setting **total memory** and other configuration parameters.

Consider changing the value of the **total memory** configuration parameter:

- When you change the amount of RAM on your machine
- When the pattern of use of your machine changes
- If you allocate memory for **additional network memory** for Adaptive Server

If Adaptive Server Cannot Start

When Adaptive Server starts, it must acquire the full amount of memory set by **total memory** from the operating system. If Adaptive Server cannot start for this reason, reduce the memory requirements for Adaptive Server by editing the value of the **total memory** parameter in the server's configuration file. You may also need to reduce the values for other configuration parameters that require large amounts of memory. Then restart Adaptive Server so that it will use the new values. See Chapter 11, "Setting Configuration Parameters," for information about using configuration files.

How Adaptive Server Uses Memory

The value of the **total memory** parameter specifies the total amount of memory that Adaptive Server requires at start-up. For example, if the **total memory** parameter has a value of 50,000 pages, Adaptive Server tries to obtain 97.65MB (50,000 * 2048) of memory at start-up. If this amount is not available, Adaptive Server will not start.

When Adaptive Server starts, it allocates memory for:

- Adaptive Server executable code

- Memory used by Adaptive Server for nonconfigurable data structures
- Memory for user-configurable parameters

Adaptive Server then divides the remaining memory between the data cache and the procedure cache, based on the value of the **procedure cache percent** parameter. Figure 8-1 illustrates how Adaptive Server allocates memory.

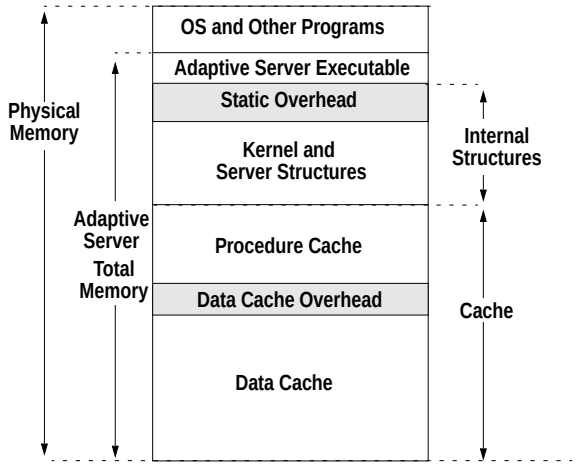


Figure 8-1: Example of memory allocation

When you configure Adaptive Server parameters that require memory, this memory reduces the amount of memory available to the procedure and data cache pools. Figure 8-2 shows how configuring one parameter that uses memory, in this case **number of worker processes**, reduces the memory available to the procedure and data cache pools.

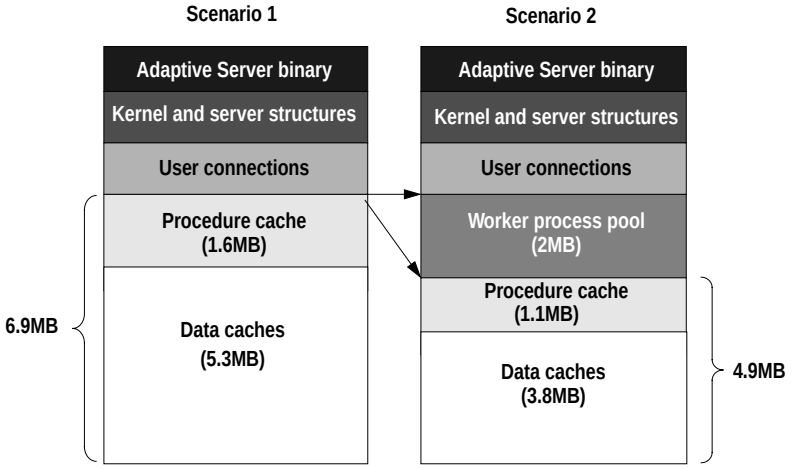


Figure 8-2: How changing configuration parameters reduces cache size

The size of the data and procedure caches has a significant impact on overall performance. On a development system, you may want to increase the amount of memory dedicated to the procedure cache. On a production system, however, you may want to reduce the size of the procedure cache to provide more memory for the data cache. See Chapter 16, "Memory Use and Performance," in the *Performance and Tuning Guide* for recommendations on optimizing procedure cache size.

To determine the amount of memory available for caches, you can estimate the amount of overhead required for Adaptive Server and subtract that amount from the **total memory.** Or you can directly compute the size of the caches, using start-up messages from the error log file.

System Procedures for Configuring Memory

The three system procedures that you need to use while configuring Adaptive Server memory are:

- **sp_configure**
- **sp_helpconfig**
- **sp_monitorconfig**

Using *sp_configure* to Set Configuration Parameters

The full syntax and usage of **sp_configure** and details on each configuration parameter are covered in Chapter 11, "Setting Configuration Parameters." The rest of this chapter discusses issues pertinent to configuring the parameters that use Adaptive Server memory. Execute **sp_configure**, specifying the "Memory Use" group to see these parameter settings on your server.

```
sp_configure "Memory Use"
```

Parameter Name	Default	Memory Used	Config Value	Run Value
additional network memory	0	48	49152	49152
allow resource limits	0	#6	1	1
audit queue size	100	42	100	100
default network packet size	512	#191	512	512
disk i/o structures	256	21	256	256
enable rep agent threads	0	0	0	0
event buffers per engine	100	#59	100	100
executable codesize + overhead	0	14540	0	14540
max cis remote servers	25	0	25	25
max number network listeners	5	82	1	1
max online engines	1	895	6	6
max roles enabled per user	20	#10	20	20
memory per worker process	1024	51	1024	1024
number of alarms	40	7	40	40
number of aux scan descriptors	200	#99	200	200
number of devices	10	#35	80	80
number of languages in cache	3	4	3	3
number of large i/o buffers	6	241	15	15
number of locks	5000	5274	50000	50000
number of mailboxes	30	2	100	100
number of messages	64	36	1500	1500
number of open databases	12	434	12	12
number of open indexes	500	63	150	150
number of open objects	500	85	150	150
number of remote connections	20	33	20	20
number of remote logins	20	6	5	5
number of remote sites	10	245	3	3
number of user connections	25	4885	60	60
number of worker processes	0	4097	50	50
partition groups	1024	21	1024	1024
permission cache entries	15	#65	15	15
procedure cache percent	20	11600	18	18
remote server pre-read packets	3	#32	3	3
stack guard size	4096	#516	4096	4096
stack size	34816	#4387	34816	34816
total data cache size	0	51019	0	51019
total memory	12000	94000	47000	47000

A "#" in the "Memory Used" column indicates that this parameter is a component of another parameter and that its memory use is included in the memory use for the other component. For example, memory used for **stack size** and **stack guard size** contributes to the memory requirements for each user connection and worker process, so the value is included in the memory required for **number of user connections** or that for **number of worker processes**.

Some of the values in this list are computed values. They cannot be set directly with **sp_configure**, but are reported to show where memory is allocated. Among the computed values is **total data cache size**.

Using *sp_helpconfig* to Get Help on Configuration Parameters

The system procedure **sp_helpconfig** estimates the amount of memory required for a given configuration parameter and value. It also provides a short description of the parameter, information about the minimum, maximum, and default values for the parameter, the run value, and the amount of memory used at the current run value. **sp_helpconfig** is particularly useful if you are planning substantial changes to a server, such as loading large, existing databases from other servers, and you want to estimate how much memory is needed.

To see how much memory is required to configure a parameter, enter enough of the parameter name so that it is a unique name and the value you want to configure:

```
sp_helpconfig "worker processes", "50"
```

```
number of worker processes is the maximum number of worker processes
that can be in use Server-wide at any one time.
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
            0    2147483647             0             0           0

Configuration parameter, 'number of worker processes', will consume
4515K of memory if configured at 50.
```

You can also use **sp_helpconfig** to determine the value to use for **sp_configure**, if you know how much memory you want to allocate to a specific resource:

```
sp_helpconfig "user connections", "5M"
```

```
number of user connections sets the maximum number of user connections
that can be connected to SQL Server at one time.
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
            5    2147483647            25            25       1982

Configuration parameter, 'number of user connections', can be configured
to 64 to fit in 5M of memory.
```

The important difference between the syntax of these two statements is the use of a unit of measure in the second example to indicate to the procedure that the value is a size, not a configuration value. The valid units of measure are:

- **P** – Pages, (Adaptive Server 2K pages)
- **K** – Kilobytes
- **M** – Megabytes
- **G** – Gigabytes

There are some cases where the syntax does not make sense for the type of parameter, or where Adaptive Server is not able to calculate the memory use. **sp_helpconfig** prints an error message in these cases. For example:

- Attempting to specify a size for a parameter that toggles, such as **allow resource limits**
- Attempting to specify a size value for a configuration parameter that is currently set to 0

For each of these conditions, and all configuration parameters that do not use memory, **sp_helpconfig** prints the message that describes the function of the parameter.

Using *sp_monitorconfig* to Find Metadata Cache Usage Statistics

The **sp_monitorconfig** system procedure displays metadata cache usage statistics on certain shared server resources. It displays the following:

- The number of databases, objects, and indexes that can be open at any one time
- The number of auxiliary scan descriptors used by referential integrity queries.
- The number of free and active descriptors
- The percentage of active descriptors

- The maximum number of descriptors used since the server was last started

For example, suppose you have configured the **number of open indexes** configuration parameter to 500. During a peak period, you can run **sp_monitorconfig** as follows to get an accurate reading of the actual metadata cache usage for index descriptors. For example:

```
sp_monitorconfig "number of open indexes"
Usage information at date and time: Aug 14 1997 8:54AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  217       283      56.60      300               No
objects
```

In this report, the maximum number of open indexes used since the server was last started is 300, even though Adaptive Server is configured for 500. Therefore, you can reset the **number of open indexes** configuration parameter to 330, to accommodate the 300 maximum used index descriptors, plus space for 10 percent more.

Major Uses of Adaptive Server Memory

This section discusses configuration parameters that use large amounts of Adaptive Server memory and those that are commonly changed at a large number of Adaptive Server installations. The parameters discussed in this section should be checked by System Administrators who are configuring a new Adaptive Server for the first time. System Administrators should review these parameters when the system configuration changes, after upgrading to a new release of Adaptive Server, or when making changes to other configuration variables that use memory.

Configuration parameters that use less memory, or that are less frequently used, are discussed in “Other Parameters That Use Memory” on page 8-13.

Adaptive Server Executable Code and Overhead

The size of the Adaptive Server executable code and overhead must be subtracted from the total memory available to the Adaptive Server process. The size of the executable code plus overhead varies by platform and release, but generally ranges from 6MB to 8MB. To determine the size of the Adaptive Server executable and overhead for your platform, use **sp_configure** to display the value of the **executable**

codesize + overhead configuration parameter. See “executable codesize + overhead” on page 11-61 for more information.

The size of this overhead changes when some configuration parameters that use memory are changed. The internal structures used are generally in three categories: overhead, kernel structures, and server structures.

In addition, when you enable Component Integration Services with the **enable cis** configuration parameter and then restart Adaptive Server, the size of the executable code and overhead increases. Other Component Integration Services configuration parameters use memory from the general pool of memory.

Data and Procedure Caches

As explained in “How Adaptive Server Uses Memory” on page 8-2, the data and procedure caches share all memory that is not dedicated to other resources within the server. Having sufficient data and procedure cache space is one of the most significant contributors to performance. This section explains the details of the split between the data cache and procedure cache and how to monitor cache sizes over time.

How Space Is Split Between Data and Procedure Cache

The proportion of the remaining memory that goes to procedure cache depends on the run value of the **procedure cache percent** configuration parameter. A value of 20 indicates that 20 percent of the total cache space is used for procedure cache and the remaining 80 percent is used for data cache.

With a **procedure cache percent** value of 20 and 5.39MB of memory available after all other memory needs have been met, the results are

- Data cache: (5.39) * (0.8) = 4.31MB (or 2207 pages)
- Procedure cache: (5.39) * (0.2) = 1.08MB (or 552 pages).

Usually, the amount of procedure cache is slightly higher than the amount indicated in this calculation because a portion of the 6 percent of miscellaneous overhead that is not used is added to the procedure cache.

Monitoring Cache Space

You can check data cache and procedure cache space with **sp_configure**:

```
sp_configure "total data cache size"
sp_configure "procedure cache percent"
```

If you are using named caches, and the total data cache size has been decreased because of increases in other configuration parameters, use **sp_cacheconfig** to monitor the size of the default data cache. As the total data cache shrinks, only 2K pool of the default cache shrinks in size; your named caches are not affected, and large I/O pools in the default data cache are not affected. You may start to experience performance problems due to increased I/O if the 2K pool in the default cache becomes too small.

Monitoring Cache Sizes Using the Errorlog

Another way to determine how Adaptive Server uses memory is to examine the memory-related messages written to the Adaptive Server error log when Adaptive Server starts. These messages state exactly how much data and procedure cache is allocated, how many **compiled objects** can reside in cache at any one time, and buffer pool sizes.

These messages provide the most accurate information regarding cache usage on Adaptive Server. As discussed earlier, the amount of memory allocated to data and procedure caches depends on the run value of the **procedure cache percent** configuration parameter.

Each of these error log messages is described below.

Procedure Cache Messages

Two error log messages provide information about the procedure cache.

```
server: Number of proc buffers allocated: 556
```

This message states the total number of procedure buffers (proc buffers) allocated in the procedure cache.

```
server: Number of blocks left for proc headers: 629
```

This message indicates the total number of procedure headers (proc headers) available for use in the procedure cache.

proc buffer

A **proc buffer** (procedure buffer) is a data structure used to manage compiled objects in the procedure cache. One proc buffer is used for every copy of a compiled object stored in the procedure cache. When Adaptive Server starts, it determines the number of proc buffers required and multiplies that value by the size of a single proc buffer (76 bytes) to obtain the total amount of memory required. It then allocates that amount of memory, treated as an array of proc buffers. Unlike some data structures, proc buffers can span pages.

proc header

A **proc header** (procedure header) is where a compiled object is stored while in the procedure cache. Depending on the size of the object to be stored, one or more proc headers may be required. The total number of compiled objects that can be stored in the procedure cache is limited by the number of available proc headers or proc buffers, whichever is less. Because stored procedures often use more than one page, the practical value of this number may be even lower.

The total size of procedure cache is the combined total of memory allocated to proc buffers (rounded up to the nearest page boundary), plus the memory allocated to proc headers.

Data Cache Messages

When Adaptive Server starts, it records the total size of each cache and the size of each pool in the cache in the error log. This example shows the default data cache with two pools and a user-defined cache with two pools:

```
Memory allocated for the default data cache cache: 8030 Kb
Size of the 2K memory pool: 7006 Kb
Size of the 16K memory pool: 1024 Kb
Memory allocated for the tuncache cache: 1024 Kb
Size of the 2K memory pool: 512 Kb
Size of the 16K memory pool: 512 Kb
```

As explained in “Monitoring Cache Space” on page 8-10, it is very important to monitor the size of the 2K memory pool in the default data cache if you are increasing the amount of memory used by other configuration parameters. The data cache error messages provide a means for a simple check when you restart Adaptive Server.

User Connections

The amount of memory required per user connection varies by platform, and it changes when you change other configuration variables, including:

- **default network packet size**
- **stack size** and **stack guard size**
- **user log cache size**
- **max roles enabled per user**

Changing any of these configuration parameters changes the amount of space used by each user connection: You have to multiply the difference in size by the number of user connections. For example, if you have 300 user connections, and you are considering increasing the **stack size** from 34K to 40K, the new value requires 1800K more memory, reducing the size of the data and procedure caches.

Open Databases, Open Indexes, and Open Objects

The three configuration parameters that control the total number of databases, indexes, and objects that can be open at one time are managed by special caches called **metadata caches**. The metadata caches reside in the kernel and server structures portion of Adaptive Server memory. You configure space for each of these caches with the following parameters:

- **number of open databases**
- **number of open indexes**
- **number of open objects**

When Adaptive Server opens a database or accesses an index or an object, it needs to read information about it in the corresponding system tables: *sysdatabases*, *sysindexes*, and *sysobjects*. The metadata caches for databases, indexes, or objects let Adaptive Server access the information that describes it in the *sysdatabases*, *sysindexes*, or *sysobjects* row directly in its in-memory structure. This improves performance because Adaptive Server bypasses expensive calls that require disk access. It also reduces synchronization and spinlock contention when Adaptive Server has to retrieve database, index, or object information at run time.

Managing individual metadata caches for databases, indexes, or objects is beneficial for a database that contains a large number of indexes and objects and where there is high concurrency among users. For more information about configuring the number of metadatacaches, see “number of open databases” on page 11-62, “number of open indexes” on page 11-64, and “number of open objects” on page 11-66.

Number of Locks

All processes in Adaptive Server share a pool of lock structures. As a first estimate for configuring the number of locks using the **number of locks** configuration parameter, multiply the number of concurrent user connections you expect, **plus** the **number of worker processes** that you have configured, by 20. The number of locks required by queries can vary widely. See “number of locks” on page 11-58 for more information. For information on how worker processes use memory, see “Worker Processes” on page 8-14.

Database Devices and Disk I/O Structures

The **number of devices** configuration parameter controls the number of database devices that can be used by Adaptive Server for storing data. See “number of devices” on page 11-38 for more information.

When a user process needs to perform a physical I/O, the I/O is queued in a disk I/O structure. See “disk i/o structures” on page 11-37 for more information.

Other Parameters That Use Memory

This section discusses configuration parameters that use moderate amounts of memory or are infrequently used.

Parallel Processing

Parallel processing requires more memory than serial processing. The configuration parameters that affect parallel processing are:

- **number of worker processes**
- **memory per worker process**
- **partition groups**

- **number of mailboxes** and **number of messages**

Worker Processes

The configuration parameter **number of worker processes** sets the total number of worker processes available at one time in Adaptive Server. Each worker process requires about the same amount of memory as a user connection.

Changing any of the following parameters changes the amount of memory required for each worker process:

- **default network packet size**
- **stack size** and **stack guard size**
- **user log cache size**
- **memory per worker process**
- **max roles enabled per user**

The **memory per worker process** configuration parameter controls the additional memory that is placed in a pool for all worker processes. This additional memory stores miscellaneous data structure overhead and inter-worker process communication buffers. See "Worker Process Management" on page 24-17 of the *Performance and Tuning Guide* for information on setting **memory per worker process.**

Parallel Queries and the Procedure Cache

Each worker process makes its own copy of the query plan in space borrowed from the procedure cache. The coordinating process keeps two copies of the query plan in memory.

Partition Groups

You need to reconfigure the value only if you use a very large number of partitions in the tables on your server. See "partition groups" on page 11-120 for more information.

Remote Servers

Some configuration parameters that allow Adaptive Server to communicate with other Sybase servers such as Backup Server, Component Integration Services, or XP Server use memory.

The configuration parameters that affect remote servers and that use memory are:

- **number of remote sites**
- **number of remote connections**
- **number of remote logins**
- **remote server pre-read packets**

Number of Remote Sites

Set the **number of remote sites** configuration parameter to the number of simultaneous sites you need to communicate to or from on your server. If you use only Backup Server, and no other remote servers, you can increase your data cache and procedure cache space by reducing this parameter to 1.

The connection from Adaptive Server to XP Server uses one remote site.

Other Configuration Parameters for RPCs

The following configuration parameters for remote communication use only a small amount of memory for each connection:

- **number of remote connections**
- **number of remote logins**

Each simultaneous connection from Adaptive Server to XP Server for ESP execution uses one remote connection and one remote login.

Since the **remote server pre-read packets** parameter increases the space required for each connection configured by the **number of remote connections** parameter, increasing the number of pre-read packets can have a significant effect on memory use.

Referential Integrity

If the tables in your databases use a large number of referential constraints, you may need to adjust the **number of aux scan descriptors** parameter, if user connections exceed the default setting. In most cases, the default setting is sufficient. If a user connection exceeds the current setting, Adaptive Server returns an error message suggesting that you increase the **number of aux scan descriptors** parameter setting.

Other Parameters That Affect Memory

Other parameters that affect memory are listed below. When you reset these configuration parameters, check the amount of memory they use and the effects of the change on your procedure and data cache.

- **additional network memory**
- **allow resource limits**
- **audit queue size**
- **event buffers per engine**
- **max number network listeners**
- **max online engines**
- **max SQL text monitored**
- **number of alarms**
- **number of large i/o buffers**
- **number of languages in cache**
- **permission cache entries**
-

9 Configuring Data Caches

The most common reason for administering data caches is to reconfigure them for performance. This chapter is primarily concerned with the **mechanics** of working with data caches. Chapter 16, "Memory Use and Performance," in the *Performance and Tuning Guide* discusses performance concepts associated with data caches.

This chapter describes how to create and administer named caches on Adaptive Server. Topics include:

- The Data Cache on Adaptive Server 9-1
- Cache Configuration Commands 9-3
- Information on Data Caches 9-4
- Configuring Data Caches 9-6
- Dividing a Data Cache into Memory Pools 9-11
- Binding Objects to Caches 9-15
- Getting Information About Cache Bindings 9-17
- Dropping Cache Bindings 9-19
- Changing the Wash Area for a Memory Pool 9-20
- Resizing Named Data Caches 9-24
- Dropping Data Caches 9-26
- Changing the Size of Memory Pools 9-27
- Dropping a Memory Pool 9-29
- Cache Binding Effects on Memory and Query Plans 9-30
- Configuring Data Caches with the Configuration File 9-31
- Cache Configuration Guidelines 9-35

The Data Cache on Adaptive Server

The data cache holds the data, index, and log pages currently in use as well as pages used recently by Adaptive Server. When you first install Adaptive Server, it has a single default data cache that is used for all data, index, and log activity. You can divide this cache by creating named data caches. Also, you can create pools within the named caches and the default cache to perform large I/Os. You can

then bind a database, table (including the *syslogs* table), index, or text or image page chain to a named data cache.

Large I/O sizes enable Adaptive Server to perform data prefetching when the query optimizer determines that prefetching would improve performance. For example, an I/O size of 16K means that Adaptive Server can read an entire extent, or eight 2K pages, all at once, rather than performing eight separate I/Os. See Chapter 8, "Understanding the Query Optimizer," in the *Performance and Tuning Guide* for details about the optimizer.

Sorts can also take advantage of buffer pools configured for large I/O sizes.

The process of configuring named data caches divides the default cache into separate cache structures. The named data caches that you create can be used only by databases or database objects that are explicitly bound to them. All objects not explicitly bound to named data caches use the default data cache.

Adaptive Server provides user-configurable data caches in order to improve performance, especially for multiprocessor servers. A full discussion of how configuring named data caches can improve performance appears in "Named Data Caches and Performance" on page 16-12 of the *Performance and Tuning Guide*.

Figure 9-1 shows a data cache with the default cache and two named data caches. The default cache contains two pools, a 2K pool and a 16K pool. The *User_Table_Cache* cache has a 2K pool and a 16K pool. The *Log_Cache* has a 2K pool and a 4K pool.

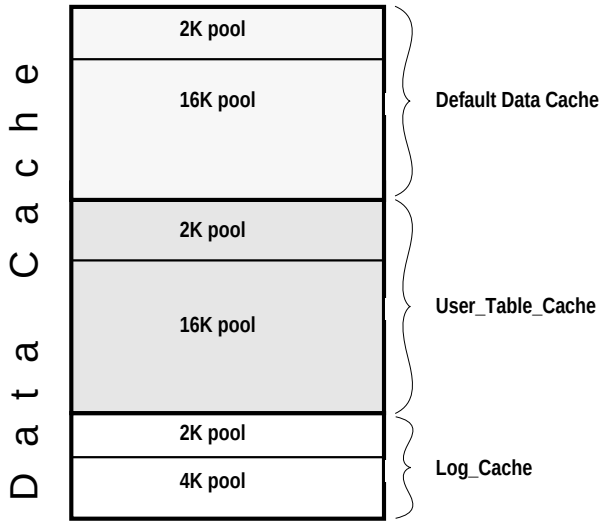


Figure 9-1: Data cache with default cache and two named data caches

Cache Configuration Commands

Table 9-1 lists commands for configuring named data caches, for binding and unbinding objects to caches, and for reporting on cache bindings. It also lists procedures that you might use to check the size of your database objects, and commands that control cache usage at the object, command, or session level.

Table 9-1: Procedures and commands for using named caches

Command	Function
sp_cacheconfig	Creates or drops named caches, and changes the size, cache type, or cache policy.
sp_poolconfig	Creates and drops I/O pools, and changes their size, wash size, and asynchronous prefetch percent limit.
sp_bindcache	Binds databases or database objects to a cache.
sp_unbindcache	Unbinds specific objects or databases from a cache.
sp_unbindcache_all	Unbinds all objects bound to a specified cache.

Table 9-1: Procedures and commands for using named caches (continued)

Command	Function
sp_helpcache	Reports summary information about data caches and lists the databases and database objects that are bound to caches.
sp_cachestrategy	Reports on cache strategies set for a table or index, and disables or re-enables prefetching or MRU strategy.
sp_logiosize	Changes the default I/O size for the log.
sp_spaceused	Provides information about the size of tables and indexes or the amount of space used in a database.
sp_estspace	Estimates the size of tables and indexes, given the number of rows the table will contain.
sp_help	Reports which cache a table is bound to.
sp_helpindex	Reports which cache an index is bound to.
sp_helpdb	Reports which cache a database is bound to.
set showplan on	Reports on I/O size and cache utilization strategies for a query.
set statistics io on	Reports number of reads performed for a query.
set prefetch [on |off]	Enables or disables prefetching for an individual session.
select... (prefetch...lru | mru)	Forces the server to use the specified I/O size or MRU replacement strategy.

In addition to the commands to configure named data caches interactively, you can also use the configuration file. See "Configuring Data Caches with the Configuration File" on page 9-31.

Information on Data Caches

The system procedure **sp_cacheconfig** creates and configures named data caches. When you first install Adaptive Server, it has a single cache named "default data cache". To see information about caches, type:

```
sp_cacheconfig
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default       0.00 Mb     59.44 Mb
                                                ------------ ------------
                                    Total       0.00 Mb     59.44 Mb

===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 0.00 Mb,   Run Size: 59.44 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb  12174 Kb      0.00 Mb     59.44 Mb     10
```

Summary information for each cache is printed in a block at the top of the report, ending with a total size for all configured caches. Then, for each cache, there is a block of information reporting the configuration for the memory pools in the cache.

The meanings of the columns in the block of output describing caches are as follows:

- "Cache Name" gives the name of the cache.
- "Status" indicates whether the cache is active. Possible values are:
  - "Pend / Act" – the cache was just created and will be active after a restart.
  - "Active" – the cache is currently active.
  - "Pend / Del" – the cache is active, but will be deleted at the next restart of the server. The cache size was reset to 0 interactively.
- "Type" indicates whether the cache can store data and log pages ("Mixed") or log pages only ("Log Only"). Only the default cache has the type "Default." You cannot change the type of the default data cache or change the type of any other cache to "Default."
- "Config Value" displays the size of the cache after the next restart of Adaptive Server. In the preceding example output, the default data cache has not been explicitly configured, so its size is 0.
- "Run Value" displays the size that Adaptive Server is currently using. For the default data cache, this size is always the amount of all data cache space that is not explicitly configured to another cache.

The second block of output begins with three lines of information that describe the cache. The first two lines repeat information from the summary block at the top. On the third line, "Config

Replacement" and "Run Replacement" show the cache policy, which is either "strict LRU" or "relaxed LRU". The run setting is the setting in effect; if the policy has been changed since the server was restarted, the config setting will be different from the run setting.

sp_cacheconfig then provides a row of information for each pool in the cache:

- "IO Size" shows the size of the buffers in the pool. When you first configure a cache, all the space is assigned to the 2K pool. Valid sizes are 2K, 4K, 8K, and 16K.
- "Wash Size" indicates the wash size for the pool. See "Changing the Wash Area for a Memory Pool" on page 9-20.
- "Config Size" and "Run Size" display the configured size and the size currently in use. These differ for the 2K pool because you cannot explicitly configure its size. These may differ for other pools if you have tried to move space between them, and some of the space could not be freed.
- "APF Percent" displays the percentage of the pool that can hold unused buffers brought in by asynchronous prefetch.

A summary line prints the total size of the cache or caches displayed.

Configuring Data Caches

After all other memory needs on Adaptive Server have been satisfied, all remaining space is available for the data cache. The first step in planning cache configuration and implementing caches is to set the **total memory** configuration parameter. After you set the configuration parameter and restart Adaptive Server, you can see exactly how much space is available for data caches on your server. For an overview of Adaptive Server memory usage, see Chapter 8, "Configuring Memory."

You can configure data caches in two ways:

- Interactively, using **sp_cacheconfig** and **sp_poolconfig**
- By editing your configuration file

The following sections describe the process of cache configuration using **sp_cacheconfig** and **sp_poolconfig**. See "Configuring Data Caches with the Configuration File" on page 9-31 for information about cache configuration using the configuration file.

Each time you execute **sp_cacheconfig** or **sp_poolconfig**, Adaptive Server writes the new cache or pool information into the configuration file

and copies the old version of the file to a backup file. A message giving the backup file name is sent to the error log.

The syntax to create a new cache is:

sp_cacheconfig *cache_name*, "*size*[P|K|M|G]"

Size units can be specified with:

- **P** – Pages, (Adaptive Server 2K pages)
- **K** – Kilobytes (default)
- **M** – Megabytes
- **G** – Gigabytes

Maximum data cache size is limited only by the amount of memory available on your system.

This command configures a 10MB cache named *pubs_cache*:

```
sp_cacheconfig pubs_cache, "10M"
```

This command makes changes in the system tables and writes the new values to the configuration file, but does not activate the cache. You must restart Adaptive Server for the changes to take effect.

Using **sp_cacheconfig** to see the configuration before a restart shows different "Config" and "Run" values:

```
sp_cacheconfig pubs_cache
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_cache                Pend/Act  Mixed        10.00 Mb      0.00 Mb
                                                ------------ ------------
                                    Total        10.00 Mb      0.00 Mb
```

The status "Pend / Act" for *pubs_cache* shows that the configuration of this cache is pending, waiting for a restart. "Config Value" displays 10MB, and "Run Value" displays the value 0. Run values and configuration values are also different when you delete caches and when you change their size.

The section of output that provides detail about pools is not printed for caches that are not active.

After a restart of Adaptive Server, **sp_cacheconfig** reports:

```
sp_cacheconfig
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default       0.00 Mb     49.37 Mb
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                                ------------ ------------
                                      Total      10.00 Mb     59.37 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 0.00 Mb,   Run Size: 49.37 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb  10110 Kb      0.00 Mb     49.37 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

The *pubs_cache* is now active, and all the space is assigned to the 2K pool. The size of the default cache has been reduced by 10MB. The remainder of the difference in the size of the default cache and the total amount of cache available is due to changing overhead values. See “How Overhead Affects Total Cache Space” on page 9-18 for examples.

You can create as many caches as you want before restarting Adaptive Server. You must restart Adaptive Server before you can configure pools or bind objects to newly created caches.

Explicitly Configuring the Default Cache

If you want to “lock in” some portion of the cache space for the default data cache, you can execute **sp_cacheconfig** with **default data cache** and a size value. This command ensures that no other cache configuration commands reduce the size of the default cache to less than 25MB:

```
sp_cacheconfig "default data cache", "25M"
```

After a restart of the server, “Config Value” shows the value.

```
sp_cacheconfig
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     49.37 Mb
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                                ------------ ------------
                                      Total      10.00 Mb     59.37 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 49.37 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb  10110 Kb     00.00 Mb     49.37 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

This command sets a minimum size for the default data cache. You can change the minimum, but you cannot inadvertently allocate this space to other caches. With a minimum default data cache size set, the "Run Value" still shows that the default data cache is allocated all of the memory not explicitly allocated to other caches.

➤ ***Note***

If you "lock in" space in the default data cache and then reduce the amount of memory available below that level, Adaptive Server will not start. Both editing your configuration file to increase the size of other caches and increasing the values of configuration parameters that require memory can reduce the size of the default data cache. See Chapter 8, "Configuring Memory," for information on configuration parameters that require memory.

You might want to lock in memory for the default cache as a protection for performance, which depends on enough space being available in the cache for data and index pages. Many configuration parameters use memory, and when the configuration values for these parameters is increased, the result is a reduction in space for the 2K pool in the default data cache. All other cache space is explicitly configured with **sp_cacheconfig**, and other pools in the default data cache are explicitly configured with **sp_poolconfig**.

Changing a Cache's Type

If you want to reserve a cache for use only by the transaction log, change the cache's type to "logonly". The following example creates the cache *pubs_log* with the type "logonly":

```
sp_cacheconfig pubs_log, "7M", "logonly"
```

The following shows the state of the cache before a restart:

```
Cache Name          Status    Type     Config Value Run Value
------------------- --------- -------- ------------ ------------
pubs_log            Pend/Act  Log Only      7.00 Mb      0.00 Mb
                                       ------------ ------------
                                Total       7.00 Mb      0.00 Mb
```

You can change the type of an existing "mixed" cache, as long as no non-log objects are bound to it:

```
sp_cacheconfig pubtune_cache, logonly
```

➤ ***Note***

In high transaction environments, Adaptive Server usually performs best with a 4K pool configured for the transaction log. For information on configuring caches for improved log performance, see "Matching Log I/O Size for Log Caches" on page 9-14.

Configuring Cache Replacement Policy

If a cache is dedicated to a table or an index, and the cache has little or no buffer replacement once the system reaches a stable state, you can set relaxed LRU (least recently used) replacement policy. Relaxed LRU replacement policy can improve performance for caches where there is little or no buffer replacement occurring. Relaxed LRU replacement policy can also improve performance for most log caches. See Chapter 16, "Memory Use and Performance," in the *Performance and Tuning Guide* for more information. To set relaxed replacement policy, use:

```
sp_cacheconfig pubs_log, relaxed
```

The default value is "strict".

You can create a cache and specify its cache type and the replacement policy in one command:

```
sp_cacheconfig pubs_log, "3M", logonly, relaxed
sp_cacheconfig pubs_cache, "10M", mixed, strict
```

You must restart Adaptive Server for cache replacement policy changes to take effect. Here are the results after a restart:

```
sp_cacheconfig
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
pubs_log                  Active    Log Only      7.00 Mb      7.00 Mb
                                                ------------ ------------
                                      Total      42.00 Mb     59.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   1432 Kb      0.00 Mb      7.00 Mb     10
```

Dividing a Data Cache into Memory Pools

After you create a data cache, you can divide it into memory pools, each with a different I/O size. In any cache, you can have only one pool of each I/O size.

When Adaptive Server performs large I/Os, multiple pages are read into the cache at the same time. These pages are always treated as a unit: They age in the cache and are written to disk as a unit.

By default, when you create a named data cache, all of its space is assigned to the default 2K memory pool. Creating additional pools reassigns some of that space to other pools, reducing the size of the

2K pool. For example, if you create a data cache with 50MB of space, all the space is assigned to the 2K pool. If you configure a 4K pool with 30MB of space in this cache, the 2K pool is reduced to 20MB.

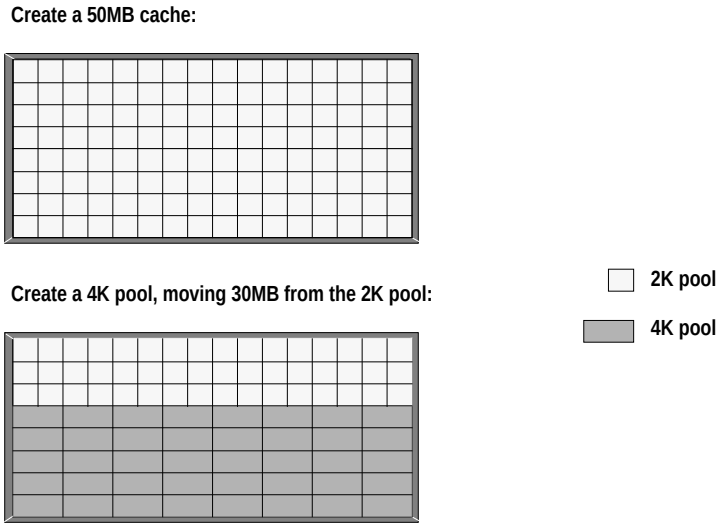


Figure 9-2: Configuring a cache and a 4K memory pool

Once you create the pools, you can move space between them. For example, in a cache with a 20MB 2K pool and a 30MB 4K pool, you can configure a 16K pool, taking 10MB of space from the 4K pool.

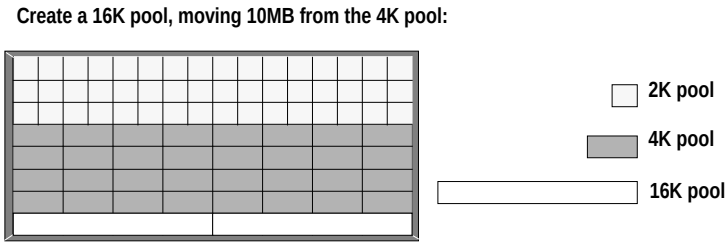


Figure 9-3: Moving space from an existing pool to a new pool

The commands that move space between pools within a cache do not require a restart of Adaptive Server to take effect, so you can

reconfigure pools to meet changing application loads with little impact on server activity.

In addition to creating pools in the caches you configure, you can add memory pools for I/Os up to 16K to the default data cache.

The syntax for configuring memory pools is:

```
sp_poolconfig cache_name, "memsize[P|K|M|G]",
   "config_poolK" [, "affected_poolK"]
```

Pool configuration always configures the *config_pool* to the size specified in the command. It always affects a second pool (the *affected_pool*) by moving space to or from that pool. If you do not specify the *affected_pool*, the space is taken from or allocated to the 2K pool. The minimum size for a pool is 512K.

This example creates a 7MB pool of 16K pages in the *pubs_cache* data cache:

```
sp_poolconfig pubs_cache, "7M", "16K"
```

This command reduces the size of the 2K memory pool. To see the current configuration, run **sp_cacheconfig**, giving only the cache name:

```
sp_cacheconfig pubs_cache
```

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_cache               Active    Mixed        10.00 Mb     10.00 Mb
                                                ------------ ------------
                                     Total      10.00 Mb     10.00 Mb
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb      3.00 Mb     10
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

You can also create memory pools in the default data cache.

In the following example, you start with this cache configuration:

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
                                                ------------ ------------
                                      Total      25.00 Mb     42.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
```

This command creates a 16K pool in the default data cache:

```
sp_poolconfig "default data cache", "8M", "16K"
```

It results in this configuration, reducing the "Run Size" of the 2K pool:

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
                                                ------------ ------------
                                      Total      25.00 Mb     42.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     34.29 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

You do not need to configure the size of the 2K memory pool in caches that you create. Its "Run Size" represents all the memory not explicitly configured to other pools in the cache.

Matching Log I/O Size for Log Caches

If you create a cache for the transaction log of a database, configure most of the space in that cache to match the log I/O size. The default value is 4K, but Adaptive Server uses 2K I/O for the log if a 4K pool is not available. The log I/O size can be changed with the system procedure **sp_logiosize**. The log I/O size of each database is reported in the error log when Adaptive Server starts, or you can check the size of a database by using the database and issuing **sp_logiosize** with no parameters.

This example creates a 4K pool in the *pubs_log* cache:

```
sp_poolconfig pubs_log, "3M", "4K"
```

You can also create a 4K memory pool in the default data cache for use by transaction logs of any databases that are not bound to another cache:

```
sp_poolconfig "default data cache", "2.5M", "4K"
```

See Chapter 16, "Choosing the I/O Size for the Transaction Log," in the *Performance and Tuning Guide* for information on tuning the log I/O size.

Binding Objects to Caches

The system procedure **sp_bindcache** assigns a database, table, index or text/image object to a cache. Before you can bind an entity to a cache, the following conditions must be met:

- The named cache must exist, and its status must be "Active."
- The database or database object must exist.
- To bind tables, indexes, or objects, you must be using the database where they are stored.
- To bind system tables, including the transaction log table *syslogs*, the database must be in single-user mode.
- To bind a database, you must be using *master*, and the database must be in single user mode.
- To bind a database, user table, index, text object, or image object to a cache, the type of cache must be "Mixed." Only the *syslogs* table can be bound to a cache of "Log Only" type.
- You must own the object or be the Database Owner or the System Administrator.

You must restart Adaptive Server after creating caches in order to bind objects to them. Bindings take effect immediately and do not require a restart.

The syntax for binding objects to caches is:

```
sp_bindcache cache_name, dbname [,[owner.]tablename
   [, indexname |  "text only" ] ]
```

The owner name is optional if the table is owned by "dbo".

This command binds the *titles* table to the *pubs_cache*:

```
sp_bindcache  pubs_cache, pubs2, titles
```

To bind an index on *titles*, add the index name as the third parameter:

```
sp_bindcache pubs_cache, pubs2,  titles, titleind
```

The owner name is not needed in the examples above because the objects in the *pubs2* database are owned by "dbo". To specify a table owned by any other user, add the owner name. You must enclose the parameter in quotation marks, since the period in the parameter is a special character:

```
sp_bindcache pubs_cache, pubs2, "fred.sales_east"
```

This command binds the transaction log, *syslogs*, to the *pubs_log* cache:

```
sp_bindcache pubs_log, pubs2, syslogs
```

The database must be in single-user mode before you can bind any system tables, including the transaction log, *syslogs*, to a cache. Use the **sp_dboption** system procedure from *master*, and a **use *database*** command, and run **checkpoint**:

```
sp_dboption pubs2, single, true

use pubs2

checkpoint
```

text and *image* columns for a table are stored in a separate data structure in the database. To bind this object to a cache, add the "text only" parameter:

```
sp_bindcache pubs_cache, pubs2, au_pix,
"text only"
```

This command, executed from *master*, binds the *tempdb* database to a cache:

```
sp_bindcache tempdb_cache, tempdb
```

You can rebind objects without dropping existing bindings.

Cache Binding Restrictions

You cannot bind or unbind a database object:

- When dirty reads are active on the object
- When a cursor is open on the object

In addition, Adaptive Server needs to lock the object while the binding or unbinding takes place, so the procedure may have a slow response time, because it waits for locks to be released. See "Locking to Perform Bindings" on page 9-30 for more information.

Getting Information About Cache Bindings

The **sp_helpcache** system procedure provides information about a cache and the entities bound to it when you provide the cache name:

```
sp_helpcache pubs_cache
```

```
Cache Name           Config Size     Run Size       Overhead
---------------------- ------------   ----------     ----------
pubs_cache            10.50 Mb       10.50 Mb        0.56 Mb

------------------ Cache Binding Information: ------------------

Cache Name      Entity Name           Type   Index Name  Status
----------      -----------           ----   ----------  ------
pubs_cache      pubs2.dbo.titles      index  titleind    V
pubs_cache      pubs2.dbo.au_pix      index  tau_pix     V
pubs_cache      pubs2.dbo.titles      table              V
pubs_cache      pubs2.fred.sales_east table              V
```

If you use **sp_helpcache** without a cache name, it prints information about all the configured caches on Adaptive Server and all the objects that are bound to them.

sp_helpcache performs string matching on the cache name, using %*cachename*%. For example, "pubs" matches both "pubs_cache" and "pubs_log".

The "Status" column reports whether a cache binding is valid ("V") or invalid ("I"). If a database or object is bound to a cache, and the cache is deleted, binding information is retained in the system tables, but the cache binding is marked as invalid. All objects with invalid bindings use the default data cache. If you subsequently create another cache with the same name, the binding becomes valid when the cache is activated by a restart of Adaptive Server.

Checking Cache Overhead

sp_helpcache can report the amount of overhead required to manage a named data cache of a given size. When you create a named data cache, all the space you request with **sp_cacheconfig** is made available for cache space. The memory needed for cache management is taken from the default data cache.

To see the overhead required for a cache, give the proposed size. You can use **P** for pages, **K** for kilobytes, **M** for megabytes, or **G** for gigabytes. The following examples check the overhead for 20,000 pages:

```
sp_helpcache "20000P"
```

```
2.08Mb of overhead memory will be needed to manage
a cache of size 20000P
```

Note that you are not wasting any cache space by configuring user caches. About 5 percent of memory is required for the structures that store and track pages in memory, whether you use a single large data cache or several smaller caches.

How Overhead Affects Total Cache Space

The example detailed in "Information on Data Caches" on page 9-4 and "Configuring Data Caches" on page 9-6 shows a default data cache with 59.44 MB of cache space available before any user-defined caches are created. When the 10MB *pubs_cache* is created and Adaptive Server is restarted, the results of **sp_cacheconfig** show a total cache size of 59.44 MB.

The process of configuring a data cache can appear to increase or decrease the total available cache. The explanation for this lies in the amount of overhead required to manage a cache of a particular size, and the fact that the overhead is not included in the values displayed by **sp_cacheconfig**.

Using **sp_helpcache** to check the overhead of the original 59.44MB default cache and the new 10MB cache shows that the change in space is due to changes in the size of overhead. The following command shows the overhead for the default data cache before any changes were made:

```
sp_helpcache "59.44M"

3.04Mb of overhead memory will be needed to manage
a cache of size 59.44M
```

This command shows the overhead for *pubs_cache*:

```
sp_helpcache "10M"

0.53Mb of overhead memory will be needed to manage
a cache of size 10M
```

The following calculations add the overhead required to manage the original cache space and then subtract the overhead for *pubs_cache*.

Original total cache size (overhead not included)	59.44
Overhead for 59.44 MB default cache	+3.04
Total cache space, including overhead	62.48
Subtract 10MB *pubs_cache* and .53MB overhead	- 10.53
Remaining space	51.95
Overhead for 51.95MB cache	- 2.69
Usable size of the default cache	49.26

Cache sizes are rounded to two places when printed by **sp_cacheconfig**, and overhead is rounded to two places by **sp_helpcache**, so you will see a small amount of rounding error in the output.

Dropping Cache Bindings

Two commands drop cache bindings:

- **sp_unbindcache** unbinds a single entity from a cache
- **sp_unbindcache_all** unbinds all objects bound to a cache

The syntax for **sp_unbindcache** is:

```
sp_unbindcache dbname [,[owner.]tablename
   [, indexname | "text only"] ]
```

This command unbinds the *pubs2* database:

```
sp_unbindcache pubs2
```

This command unbinds the *titles* table:

```
sp_unbindcache pubs2, titles
```

This commands unbinds the *titleidind* index:

```
sp_unbindcache pubs2, titles, titleidind
```

In order to unbind all the objects bound to a cache, use **sp_unbindcache_all**, giving the cache's name:

```
sp_unbindcache_all pubs_cache
```

You cannot use **sp_unbindcache_all** if more than eight databases and / or objects in eight databases are bound to the cache. You must use **sp_unbindcache** on individual databases or objects to reduce the number of databases involved to eight or less.

When you drop a cache binding for an object, all the pages currently in memory are cleared from the cache.

Changing the Wash Area for a Memory Pool

When Adaptive Server needs to read a buffer into cache, it places:

- The buffer at the LRU (least recently used) end of each memory pool, in a cache with strict LRU policy
- The buffer at the victim pointer, in a cache with relaxed LRU policy. If the recently used bit of buffer at the victim marker is set, the victim pointer is moved to the next buffer in the pool.

A portion of each pool is configured as the **wash area**. Once dirty pages (pages that have been changed in cache) pass the wash marker and enter the wash area, Adaptive Server starts an asynchronous I/O on the page. When the write completes, the page is marked clean and remains available in the cache.

The space in the wash area must be large enough so that the I/O on the buffer can complete before the page needs to be replaced. Figure 9-4 illustrates how the wash area of a buffer pool works with a strict and relaxed LRU cache:

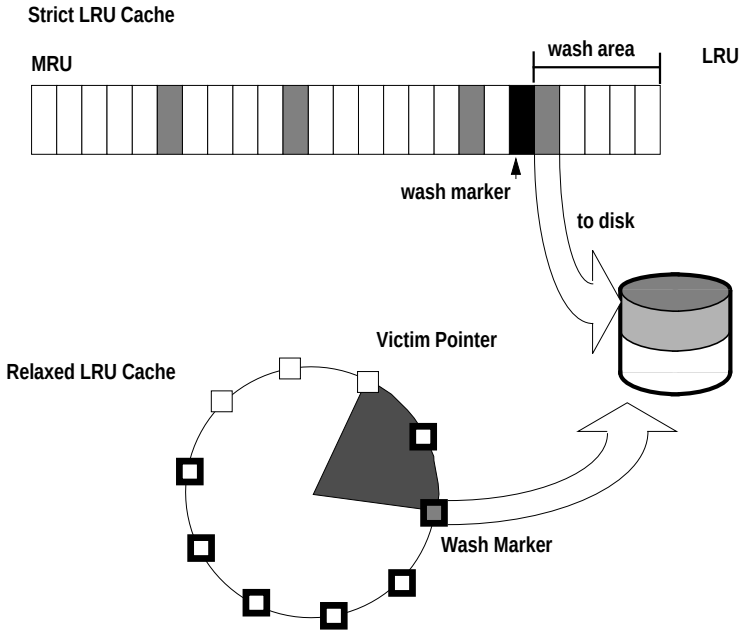


Figure 9-4: Wash area of a buffer pool

By default, the size of the wash area for a memory pool is configured as follows:

- If the pool size is less than 300MB, the default wash size is set to 20 percent of the buffers in the pool.
- If the pool size is greater than 300MB, the default wash size is 20 percent of the number of buffers in 300MB.

The minimum wash size is 10 buffers. The maximum size of the wash area is 80 percent of the pool size.

A buffer is a block of pages that matches the I/O size for the pool. Each buffer is treated as a unit: All pages in the buffer are read into cache, written to disk, and aged in the cache as a unit. For a 2K pool, 256 buffers equals 512K; for a 16K pool, 256 buffers equals 4096K.

For example, if you configure a 16K pool with 1MB of space, the pool has 64 buffers; 20 percent of 64 is 12.8. This is rounded down, so 12 buffers, or 192K, are allocated to the wash area.

When the Wash Area Is Too Small

If the wash area is too small for the usage in a buffer pool, operations that need a clean buffer may have to wait for I/O to complete on the dirty buffer at the LRU end of the pool or at the victim marker. This is called a **dirty buffer grab**, and it can seriously slow performance. Figure 9-5 shows a dirty buffer grab on a strict replacement policy cache.

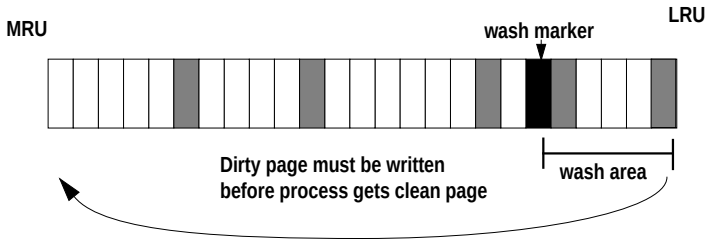


Figure 9-5: Small wash area results in a dirty buffer grab

You can use the system procedure **sp_sysmon** to determine whether dirty buffer grabs are taking place in your memory pools. Run **sp_sysmon** while the cache is experiencing a heavy period of I/O and heavy update activity, since it is the combination of many dirty pages and high cache replacement rates that usually causes dirty buffer grabs.

If the "Buffers Grabbed Dirty" output in the cache summary section shows a nonzero value in the "Count" column, check the "Grabbed Dirty" row for each pool to determine where the problem lies. Increase the size of the wash area for the affected pool. This command sets the wash area of the 2K memory pool to 720K:

```
sp_poolconfig pubs_cache, "2K", "wash=720K"
```

If the pool is very small, you may also want to increase its pool size, especially if **sp_sysmon** output shows that the pool is experiencing high turnover rates.

For more information, see "Grabbed Dirty" on page 24-79 of the *Performance and Tuning Guide.*

When the Wash Area Is Too Large

If the wash area is too large in a pool, the buffers move too quickly past the "wash marker" in cache, and an asynchronous write is

started on any dirty buffers, as shown in Figure 9-6. The buffer is marked "clean" and remains in the wash area of the MRU/LRU chain until it reaches the LRU. If another query changes a page in the buffer, Adaptive Server must perform additional I/O to write it to disk again.

If **sp_sysmon** output shows a high percentage of buffers "Found in Wash" for a strict replacement policy cache, and there are no problems with dirty buffer grabs, you may want to try reducing the size of the wash area. See "Found in Wash" on page 24-77 of the *Performance and Tuning Guide* for more information.

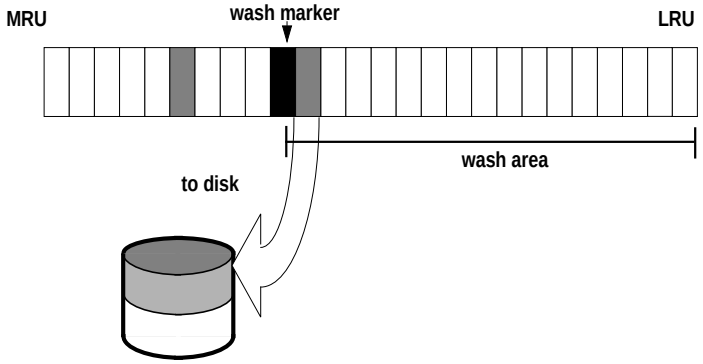


Figure 9-6: Effects of making the wash area too large

Changing the Asynchronous Prefetch Limit for a Pool

The asynchronous prefetch limit specifies the percentage of the pool that can be used to hold pages that have been brought into the cache by asynchronous prefetch, but have not yet been used by any queries. The default value for the server is set with the configuration parameter **global async prefetch limit**. Pool limits, set with **sp_poolconfig**, override the default limit for a single pool.

This command sets the percentage for the 2K pool in the *pubs_cache* to 20:

```
sp_poolconfig pubs_cache, "2K",
   "local async prefetch limit=20"
```

Changes to the prefetch limit for a pool take effect immediately and do not require a restart of Adaptive Server. Valid values are 0–100. Setting the prefetch limit to 0 disables asynchronous prefetching in a pool. For information about the impact of asynchronous prefetch on

performance, see Chapter 18, "Tuning Asynchronous Prefetch," in the *Performance and Tuning Guide*.

Resizing Named Data Caches

To change the size of an existing cache, issue **sp_cacheconfig**, specifying a new total size for the cache. When you increase the size of a cache by specifying a larger size with **sp_cacheconfig**, all the additional space is added to the 2K pool. When you decrease the size of a cache, all the space is taken from the 2K pool. You cannot decrease the size of the 2K pool to less than 512K.

Increasing the Size of a Cache

sp_cacheconfig reports that *pubs_cache* is currently configured with 10MB of space:

```
sp_cacheconfig pubs_cache
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                               ------------ ------------
                                    Total        10.00 Mb     10.00 Mb
============================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    720 Kb      0.00 Mb      3.00 Mb     20
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

If you want to increase the size of the cache and its 2K pool, specify the new total size of the cache:

```
sp_cacheconfig pubs_cache, "20M"
```

The following output reports the configuration before a restart:

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_cache                Active    Mixed        20.00 Mb     10.00 Mb
                                                ------------ ------------
                                      Total      20.00 Mb     10.00 Mb
============================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    720 Kb      0.00 Mb      3.00 Mb     20
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

The additional 10MB has been configured and becomes available in the 2K pool at the next restart.

Decreasing the Size of a Cache

You can also reduce the size of a cache. For example, following is a report on the *pubs_log* cache:

```
sp_cacheconfig pubs_log
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      7.00 Mb      7.00 Mb
                                                ------------ ------------
                                      Total       7.00 Mb      7.00 Mb
============================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    920 Kb      0.00 Mb      4.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

The following command reduces the size of the *pubs_log* cache, reducing the size of the 2K pool:

```
sp_cacheconfig pubs_log, "6M"
```

After a restart of Adaptive Server, **sp_cacheconfig** shows:

```
ache Name               Status    Type     Config Value Run Value
----------------------- --------- -------- ------------ ------------
pubs_log                 Active    Log Only      6.00 Mb      6.00 Mb
                                                ------------ ------------
                                     Total       6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

When you reduce the size of a data cache, all the space to be removed must be available in the 2K pool. You may need to move space to the 2K pool from other pools before you can reduce the size of the data cache. In the last example, if you wanted to reduce the size of the cache to 3MB, you would need to use **sp_poolconfig** to move some memory into the 2K pool from the 4K pool. See "Changing the Size of Memory Pools" on page 9-27 for more information.

Dropping Data Caches

To completely remove a data cache, reset its size to 0:

```
sp_cacheconfig pubs_log, "0"
```

This changes the cache status to "Pend/Del". You must restart Adaptive Server for the change to take effect. Until you do, the cache remains active, and all objects bound to the cache still use it for I/O.

If you delete a data cache, and there are objects bound to the cache, the cache bindings are marked invalid at the next restart of Adaptive Server. All objects with invalid cache bindings use the default data cache. Warning messages are printed in the error log when the bindings are marked invalid. For example, if the *titles* table in the *pubs2* database is bound to a cache, and that cache is dropped, the message in the log is:

```
Cache binding for database '5', object
'208003772', index '0' is being marked invalid in
Sysattributes.
```

If you re-create the cache and restart Adaptive Server, the bindings are marked valid again.

You cannot drop the default data cache.

Changing the Size of Memory Pools

To change the size of a memory pool, use **sp_poolconfig** to specify the cache, the new size for the pool, the I/O size of the pool you want to change, and the I/O size of the pool from which the buffers should be taken. If you do not specify the final parameter, all the space is taken from or assigned to the 2K pool.

Moving Space from the 2K Memory Pool

This command checks the current configuration of the *pubs_log* cache:

```
sp_cacheconfig pubs_log
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                                ------------ ------------
                                      Total       6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

This command increases the size of the 4K pool to 5MB, moving the required space from the 2K pool:

```
sp_poolconfig pubs_log, "5M", "4K"
sp_cacheconfig pubs_log
```

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                                ------------ ------------
                                      Total       6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
```

Moving Space from Other Memory Pools

To transfer space from a pool other than the 2K pool, you specify the cache name, a "to" I/O size, and a "from" I/O size. This output shows the current configuration of the default data cache:

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                                ------------ ------------
                                      Total      25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

The following command increases the size of the 4K pool from 2.5MB to 4MB, taking the space from the 16K pool:

```
sp_poolconfig "default data cache","4M", "4K","16K"
```

This command results in the following configuration:

```
Cache Name                Status    Type     Config Value Run Value

------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                                ------------ ------------
                                      Total      25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
   16 Kb   1632 Kb      6.50 Mb      6.50 Mb     10
```

When you issue a command to move buffers between pools in a cache, Adaptive Server can move only "free" buffers. It cannot move buffers that are in use or buffers that contain changes that have not been written to disk.

When Adaptive Server cannot move as many buffers as you request, it displays an informational message, giving the requested size and the resulting size of the memory pool.

Dropping a Memory Pool

To completely remove a pool, reset its size to 0. This command removes the 16K pool and places all space in the 2K pool:

```
sp_poolconfig "default data cache", "0",  "16K"
sp_cacheconfig "default data cache"
```

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
                                                ------------ ------------
                                     Total      25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
```

If you do not specify the affected pool size (16K in the example above), all the space is placed in the 2K pool. You cannot delete the 2K pool in any cache.

When Pools Cannot Be Dropped Due to Pages In Use

If the pool you are trying to delete contains pages that are in use, or pages that have been dirtied but not written to disk, Adaptive Server moves as many pages as possible to the specified pool and prints an informational message telling you the size of the remaining pool. If the pool size is smaller than the minimum allowable pool size, you also receive a warning message saying the pool has been marked unavailable. If you run **sp_cacheconfig** after receiving on of these warnings, the pool detail section for these pools contains an extra "Status" column, with either "Unavailable / too small" or "Unavailable / deleted" for the affected pool.

You can reissue the command at a later time to complete remove the pool. Pools with "Unavailable / too small" or "Unavailable / deleted" are also removed when you restart Adaptive Server.

Cache Binding Effects on Memory and Query Plans

Binding and unbinding objects may have an impact on performance. When you bind or unbind a table or an index:

- The object's pages are flushed from the cache
- The object must be locked to perform the binding
- All query plans for procedures and triggers must be recompiled

Flushing Pages from Cache

When you bind an object or database to a cache, the object's pages that are already in memory are removed from the source cache. The next time the pages are needed by a query, they are read into the new cache. Similarly, when you unbind objects, the pages in cache are removed from the user-configured cache and read into the default cache the next time they are needed by a query.

Locking to Perform Bindings

In order to bind or unbind user tables, indexes, or text or image objects, the cache binding commands need to acquire an exclusive table lock on the object. If a user holds locks on a table, and you issue an **sp_bindcache**, **sp_unbindcache**, or **sp_unbindcache_all** on the object, the system procedure sleeps until it can acquire the locks it needs.

For databases, system tables, and indexes on system tables, the database must be in single-user mode, so there cannot be another user who holds a lock on the object.

Cache Binding Effects on Stored Procedures and Triggers

Cache bindings and I/O sizes are part of the query plan for stored procedures and triggers. When you change the cache binding for an object, all the stored procedures that reference the object are recompiled the next time they are executed. When you change the cache binding for a database, all stored procedures that reference any objects in the database that are not explicitly bound to a cache are recompiled the next time they are run.

Configuring Data Caches with the Configuration File

You can add or drop named data caches and reconfigure existing caches and their memory pools by editing the configuration file that is used when you start Adaptive Server.

➤ ***Note***

You cannot reconfigure caches and pools on a server while it is running by reading in a configuration file with **sp_configure**. Any attempt to read a configuration file that contains cache and pool configurations different from those already configured on the server causes the read to fail.

Cache and Pool Entries in the Configuration File

Each configured data cache on the server has a block of information in the configuration file:

```
[Named Cache:cache_name]
     cache size = {size | DEFAULT}
     cache status = {mixed cache | log only | default data cache}
     cache replacement policy = {DEFAULT |
         relaxed LRU replacement| strict LRU replacement }
```

Size units can be specified with:

- **P** – Pages, (Adaptive Server 2K pages)
- **K** – Kilobytes (default)
- **M** – Megabytes
- **G** – Gigabytes

This example shows the configuration file entry for the default data cache:

```
[Named Cache:default data cache]
      cache size = DEFAULT
      cache status = default data cache
      cache replacement policy = strict LRU replacement
```

The default data cache entry is the only cache entry that is required in order for Adaptive Server to start. It must have the cache size and cache status, and the status must be “default data cache.”

If the cache has pools configured in addition to the 2K pool, the block in the preceding example is followed by a block of information for each pool:

```
[16K I/O Buffer Pool]
        pool size = size
        wash size = size
        local async prefetch limit = DEFAULT
```

➤ ***Note***

In some cases, there is no configuration file entry for the 2K pool in a cache. If you change the asynchronous prefetch percentage with **sp_poolconfig**, the change is not written to the configuration file, only to system tables.

The following example shows output from **sp_cacheconfig**, followed by the configuration file entries that match this cache and pool configuration:

```
 Cache Name                Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
pubs_cache               Active    Mixed        20.00 Mb     20.00 Mb
pubs_log                 Active    Log Only      6.00 Mb      6.00 Mb
tempdb_cache             Active    Mixed         4.00 Mb      4.00 Mb
                                                ------------ ------------
                                     Total      55.00 Mb     59.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 20.00 Mb,   Run Size: 20.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2662 Kb      0.00 Mb     13.00 Mb     10
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
```

```
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
============================================================================
Cache: tempdb_cache,   Status: Active,   Type: Mixed
     Config Size: 4.00 Mb,   Run Size: 4.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    818 Kb      0.00 Mb      4.00 Mb     10
```

This is the matching configuration file information:

```
[Named Cache:default data cache]
        cache size = 25M
        cache status = default data cache
        cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
        pool size = 6656.0000k
        wash size = 3844 K
        local async prefetch limit = DEFAULT

[4K I/O Buffer Pool]
        pool size = 4.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_cache]
        cache size = 20M
        cache status = mixed cache
        cache replacement policy = strict LRU replacement

[16K I/O Buffer Pool]
        pool size = 7.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_log]
        cache size = 6M
        cache status = log only
        cache replacement policy = relaxed LRU replacement

[4K I/O Buffer Pool]
        pool size = 5.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT
```

```
[Named Cache:tempdb_cache]
        cache size = 4M
        cache status = mixed cache
        cache replacement policy = DEFAULT
```

For more information about the configuration file, see Chapter 11, "Setting Configuration Parameters."

◆ ***WARNING!***

Be sure to check the total memory configuration parameter and allow enough memory for other Adaptive Server needs. If you attempt to assign too much memory to data caches in your configuration file, Adaptive Server will not start. If this occurs, edit the configuration file to reduce the amount of space in the data caches, or increase the total memory allocated to Adaptive Server. See Chapter 8, "Configuring Memory," for suggestions on monitoring cache sizes.

Configuration File Errors

If you edit your configuration file by hand, check the cache, pool, and wash sizes carefully. Certain configuration file errors can cause start-up failure:

- The total size of all of the caches cannot be greater than the amount of total memory, minus other Adaptive Server memory needs.
- The total size of the pools in any cache cannot be greater than the size of the cache.
- The wash size cannot be too small (less than 20 percent of the pool size, with a minimum of 10 buffers) and cannot be larger than 80 percent of the buffers in the pool.
- The default data cache status must be "default data cache", and the size must be specified, either as a numeric value or as "DEFAULT".
- The status and size for any cache must be specified.
- The pool size and wash size for all pools larger than 2K must be specified.

- The status of all user-defined caches must be “mixed cache” or “log only”.
- The cache replacement policy and the asynchronous prefetch percentage are optional, but, if specified, they must have correct parameters or “DEFAULT”.

In most cases, problems with missing entries are reported as “unknown format” errors on lines immediately following the entry where the size, status, or other information was omitted. Other errors provide the name of the cache where the error occurred and the type of error. For example, you see this error if the wash size for a pool is specified incorrectly:

```
The wash size for the 4k buffer pool in cache
pubs_cache has been incorrectly configured. It
must be a minimum of 10 buffers and a maximum of
80 percent of the number of buffers in the pool.
```

Cache Configuration Guidelines

User-definable caches are a performance feature of Adaptive Server. This chapter addresses only the mechanics of configuring caches and pools and binding objects to caches. Performance information and suggested strategies for testing cache utilization is addressed in Chapter 16, “Memory Use and Performance,” in the *Performance and Tuning Guide*. Following are some general guidelines for System Administrators:

- Make sure that your default data cache is large enough for all cache activity on unbound tables and indexes. All objects that are not explicitly bound to a cache use the default cache. This includes any unbound system tables in the user databases, the system tables in master, and any other objects that are not explicitly bound to a cache.
- During recovery, only the 2K memory pool of the default cache is active. Transactions logs are read into the 2K pool of the default cache. All transactions that must be rolled back or rolled forward must read data pages into the default data cache. If the default data cache is too small, it can slow recovery time.
- Do not “starve” the 2K pool in any cache. For many types of data access, there is no need for large I/O. For example, a simple query that uses an index to return a single row to the user might use 4 or 5 2K I/Os, and gain nothing from 16K I/O.

- Certain commands can perform only 2K I/O: **disk init**, certain **dbcc** commands, and **drop table**. **dbcc checktable** can perform large I/O, and **dbcc checkdb** performs large I/O on tables and 2K I/O on indexes.
- For caches used by transaction logs, configure an I/O pool that matches the default log I/O size. This size is set for a database using the system procedure **sp_logiosize**. The default value is 4K.
- Trying to micro-manage every index and object and its caching can waste cache space. If you have created caches or pools that are not optimally used by the tables or indexes bound to them, they are wasting space and creating additional I/O in other caches.
- If *tempdb* is used heavily by your applications, bind it to its own cache. Note that you can bind only the entire *tempdb* database, you cannot bind individual objects from *tempdb*.
- For caches with high update and replacement rates, be sure that your wash size is large enough.
- On multi-CPU systems, spread your busiest tables and their indexes across multiple caches to avoid spinlock contention.
- Consider reconfiguring caches or the memory pools within caches to match changing workloads. Reconfiguring caches requires a restart of the server, but memory pool reconfiguration does not.

  For example, if your system performs mostly OLTP (online transaction processing) during most of the month, and has heavy DSS (decision support system) activity for a few days, consider moving space from the 2K pool to the 16K pool for the high DSS activity and resizing the pools for OLTP when the DSS workload ends.

10 Managing Multiprocessor Servers

This chapter provides guidelines for administering Adaptive Server on a multiprocessor. Topics include:

- Parallel Processing 10-1
- Definitions 10-1
- Target Architecture 10-2
- Configuring an SMP Environment 10-5

Parallel Processing

Adaptive Server implements the Sybase Virtual Server Architecture™, which enables it to take advantage of the parallel processing feature of symmetric multiprocessing (SMP) systems. Adaptive Server can be run as a single process or as multiple, cooperating processes, depending on the number of CPUs available and the demands placed on the server machine. This chapter describes:

- The target machine architecture for the SMP Adaptive Server
- Adaptive Server architecture for SMP environments
- Adaptive Server task management in the SMP environment
- Managing multiple engines

For information on application design for SMP systems, see Chapter 21, "How Adaptive Server Uses Engines and CPUs," in the *Performance and Tuning Guide*.

Definitions

Following are the definitions of several terms used in this chapter:

- **Process** – an execution environment scheduled onto physical CPUs by the operating system.
- **Engine** – a process running an Adaptive Server that communicates with the other Adaptive Server processes via shared memory. An engine can be thought of as one CPU's worth of processing power. It does **not** represent a particular CPU. Also referred to as a **server engine**.

- **Task** – an execution environment within the Adaptive Server that is scheduled onto engines by the Adaptive Server.
- **Affinity** – describes a process in which a certain Adaptive Server task runs only on a certain engine (**task affinity**), a certain engine handles network I/O for a certain task (**network I/O affinity**), or a certain engine runs only on a certain CPU (**engine affinity**).
- **Network affinity migration** – describes the process of moving network I/O from one engine to another. SMP systems that support this migration allow Adaptive Server to distribute the network I/O load among all of its engines.

Target Architecture

The SMP environment product is intended for machines with the following features:

- A symmetric multiprocessing operating system
- Shared memory over a common bus
- 1–32 processors
- No master processor
- Very high throughput

Adaptive Server consists of one or more cooperating processes (called **engines**), all of which run the server program in parallel. See Figure 10-1.

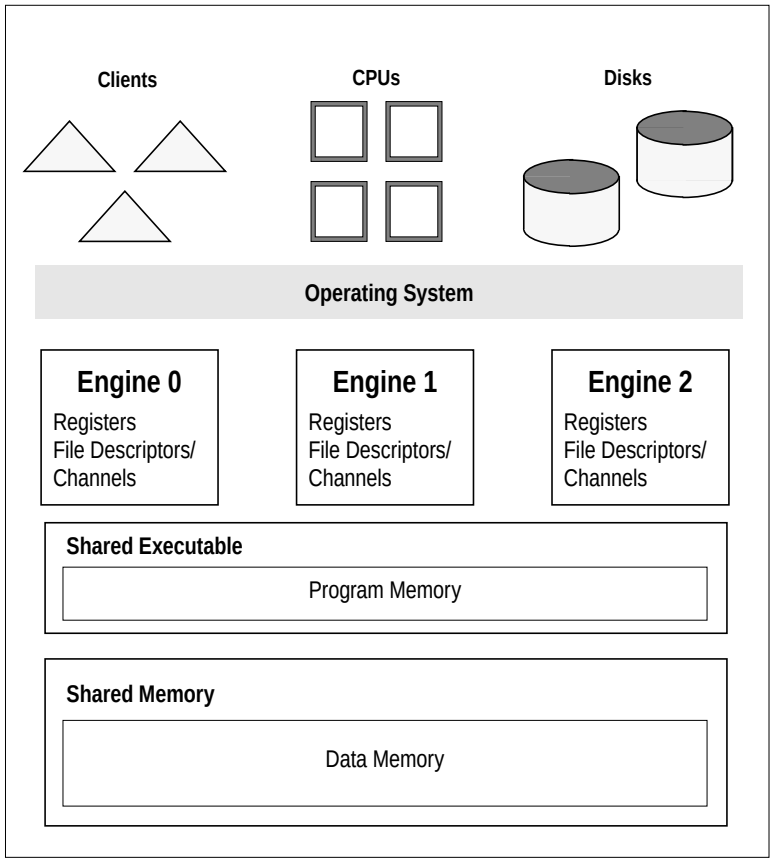


Figure 10-1: SMP environment architecture

If the SMP system supports **network affinity migration**, Adaptive Server migrates the client connection at login time to the engine that is currently servicing the smallest number of network I/O connections. Otherwise, only one of the engines, engine 0, handles the tasks involving network management. In other aspects, all engines are peers, and they communicate via shared memory.

The server engines perform all database functions, including updates and logging. Adaptive Server, not the operating system, dynamically schedules client tasks onto available engines. When an engine becomes available, it executes any runnable client task; there is no **task affinity**.

The operating system schedules the engine processes onto physical processors. Any available CPU is used for any engine; there is no **engine affinity**. The processing is called **symmetric** because the lack of affinity between processes and CPUs creates a symmetrically balanced load.

Adaptive Server Task Management for SMP

Figure 10-2 illustrates Adaptive Server task management for SMP. Here is a brief description of the process:

1. A client application issues a login request. In response, Adaptive Server creates a user task to handle work from the client.
2. The client presents Adaptive Server with work to do, that is, a series of Transact-SQL commands.
3. Adaptive Server adds the client's user task to the runnable task queue. The server engines compete for the user task at the head of the task queue.
4. The server engine that takes the user task from the queue converts the Transact-SQL commands into low-level steps such as disk I/O.
5. The engine executes each step until the task completes or blocks while waiting for I/O or locking. When the task blocks, it yields the server engine to run other user tasks. Once the block is resolved (that is, disk I/O is completed or a lock is acquired), the user task is again added to the runnable task queue.
6. After the task blocks for the last time, it continues executing until it finishes. At that time, the user task yields the server engine and moves to the sleeping task queue until the client presents the server with more work.

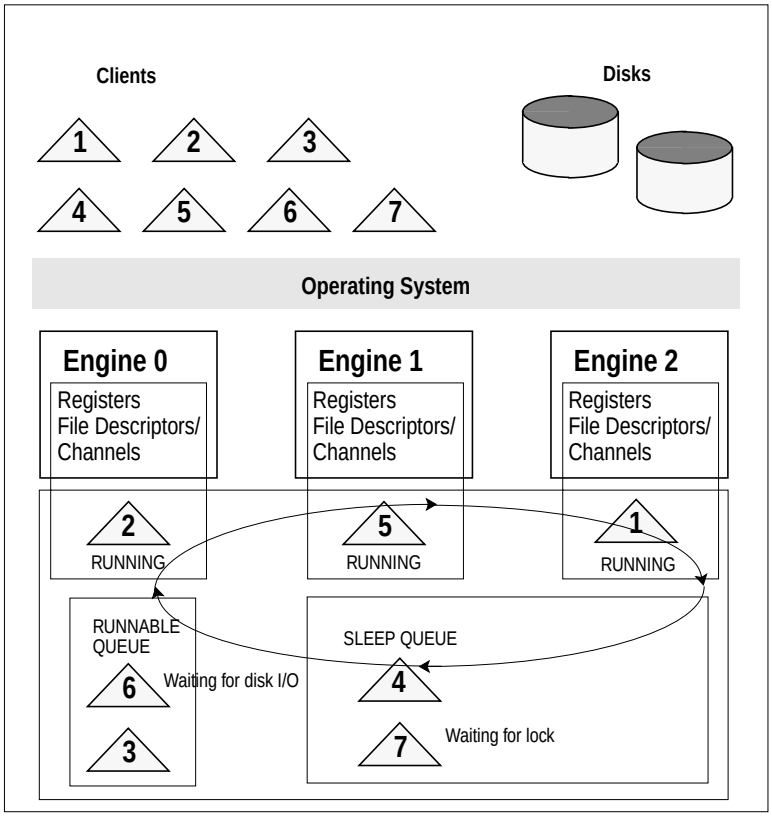


Figure 10-2: Adaptive Server task management in the SMP environment

The SMP Adaptive Server is designed in such a way that applications and users see a single database service, no matter how many engines and processors there are.

Configuring an SMP Environment

Configuring the SMP environment is much the same as configuring the uniprocessor environment, although SMP machines are typically more powerful and handle many more users. The SMP environment provides the additional ability to control the number of engines.

Managing Engines

To achieve optimum performance from an SMP system, you must maintain the right number of engines.

An engine represents a certain amount of CPU power. It is a configurable resource like memory. An engine does not represent a particular CPU.

Resetting the Number of Engines

When you create a new master device (specifically, when you run **buildmaster**), or when you start an SMP server for the first time on a database that has been upgraded from an earlier release, the system is configured for a single engine. To engage multiple engines, you must reset the number of engines the first time you start the server. You may also want to reset the number of engines at other times.

For example:

- You might want to **increase** the number of engines if current performance is not adequate for an application **and** there are enough CPUs on the machine.
- You might want to **decrease** the number of engines if a hardware failure disables CPUs on the machine.

However, increasing or decreasing engines is not a dynamic configuration, so you must restart the server to reset the number of engines.

The **max online engines** configuration parameter controls the number of engines used by Adaptive Server. Reset this parameter with the **sp_configure** system procedure. For example, to set the number of engines to 3:

1. Issue the following command:

   ```
   sp_configure "max online engines", 3
   ```

2. Stop and restart the server.

Repeat these steps whenever you need to change the number of engines. Engines other than engine 0 are brought online after recovery is complete.

Choosing the Right Number of Engines

It is important to choose the right number of engines for Adaptive Server. Here are some guidelines for choosing how many engines to use:

- **Never** have more engines than CPUs. Doing so may slow performance. If a CPU goes offline, use **sp_configure** to reduce the **max online engines** configuration parameter by 1 and restart Adaptive Server.
- Have only as many engines as you have **usable** CPUs. If there is a lot of processing by the client or other non-Adaptive Server processes, then one engine per CPU may be excessive. Remember, too, that the operating system may take up part of one of the CPUs.
- Have **enough** engines. It is good practice to start with a few engines and add engines when the existing CPUs are almost fully used. If there are too few engines, the capacity of the existing engines will be exceeded and bottlenecks may result.

Monitoring CPU Usage

To maintain the correct number of engines, monitor CPU usage with an operating system utility. See the configuration documentation for your platform for the appropriate utility for your operating system.

Managing User Connections

If the SMP system supports network affinity migration, each engine handles the network I/O for its connections. During login, Adaptive Server migrates the client connection task from engine 0 to the engine currently servicing the smallest number of connections. The client's tasks run network I/O on that engine (**network affinity**) until the connection is terminated. To determine if your SMP system supports this migration, see the configuration documentation for your platform.

By distributing the network I/O among its engines, Adaptive Server can handle more user connections. The per-process limit on the maximum number of open file descriptors no longer limits the number of connections. Adding more engines linearly increases the maximum number of file descriptors, as stored in the global variable *@@max_connections*.

As you increase the number of engines, Adaptive Server prints the increased *@@max_connections* value to standard output and the error log file after you restart the server. You can query the value as follows:

```
select @@max_connections
```

This number represents the maximum number of file descriptors allowed by the operating system for your process, minus the following file descriptors used by Adaptive Server:

- One for each master network listener on engine 0 (one for every "master" line in the interfaces file entry for that Adaptive Server)
- One for each engine's standard output
- One for each engine's error log file
- Two for each engine's network affinity migration channel
- One per engine for configuration
- One per engine for the interfaces file
- One per engine for internal use (OpenVMS only)

For example, if Adaptive Server is configured for one engine, and the value of *@@max_connections* equals 1019, adding a second engine increases the value of *@@max_connections* to 2039 (assuming only one master network listener).

You can configure **number of user connections** to take advantage of an increased *@@max_connections* limit. However, each time you decrease the number of engines using **max online engines**, you must also adjust the **number of user connections** value accordingly. Reconfiguring **max online engines** or **number of user connections** is not dynamic, so you must restart the server to change these configuration values. For information about configuring **number of user connections**, see Chapter 11, "Setting Configuration Parameters."

Managing Memory

The **total memory** configuration parameter may require special attention in SMP sites.

Not all platforms require a higher memory configuration parameter than before Adaptive Server release 11.5. If your platform does, the installed value of the **total memory** configuration parameter reflects this, so you may never need to adjust it. If error message 701:

```
There is insufficient memory to run this query
```

appears in the error log and at the client terminal, you may want to increase the amount of procedure cache available.

Configuration Parameters That Affect SMP Systems

Chapter 11, "Setting Configuration Parameters," lists configuration parameters for Adaptive Server. Some of those parameters, such as spinlock ratios, are applicable only to SMP systems.

Configuring Spinlock Ratio Parameters

Spinlock ratio parameters specify the number of internal system resources such as rows in an internal table or cache that are protected by one **spinlock**. A spinlock is a simple locking mechanism that prevents a process from accessing the system resource currently used by another process. All processes trying to access the resource must wait (or "spin") until the lock is released.

Spinlock ratio configuration parameters are meaningful only in multiprocessing systems. An Adaptive Server configured with only one engine has only one spinlock, regardless of the value specified for a spinlock ratio configuration parameter.

Table 10-1 lists system resources protected by spinlocks and the configuration parameters you can use to change the default spinlock ratio.

Table 10-1: Spinlock ratio configuration parameters

Configuration Parameter	System Resource Protected
address lock spinlock ratio	Rows in the address locks hash table
page lock spinlock ratio	Rows in the page lock hash table
table lock spinlock ratio	Rows in the table locks hash table
open index hash spinlock ratio	Index metadata descriptor hash tables
open index spinlock ratio	Index metadata descriptors
open object spinlock ratio	Object metadata descriptors
partition spinlock ratio	Rows in the internal partition caches
user log cache spinlock ratio	User log caches

The value specified for a spinlock ratio parameter defines the ratio of the particular resource to spinlocks, not the number of spinlocks. For

example, if 100 is specified for the spinlock ratio, Adaptive Server allocates one spinlock for each 100 resources. The number of spinlocks allocated by Adaptive Server depends on the total number of resources as well as on the ratio specified. The lower the value specified for the spinlock ratio, the higher the number of spinlocks.

Spinlocks are assigned to system resources in one of two ways:

- Round-robin assignment
- Sequential assignment

Round-Robin Assignment

Metadata cache spinlocks (configured by the **open index hash spinlock ratio**, **open index spinlock ratio**, and **open object spinlock ratio** parameters) use the round-robin assignment method.

Figure 10-3 illustrates one example of the round-robin assignment method and shows the relationship between spinlocks and index metadata descriptors.

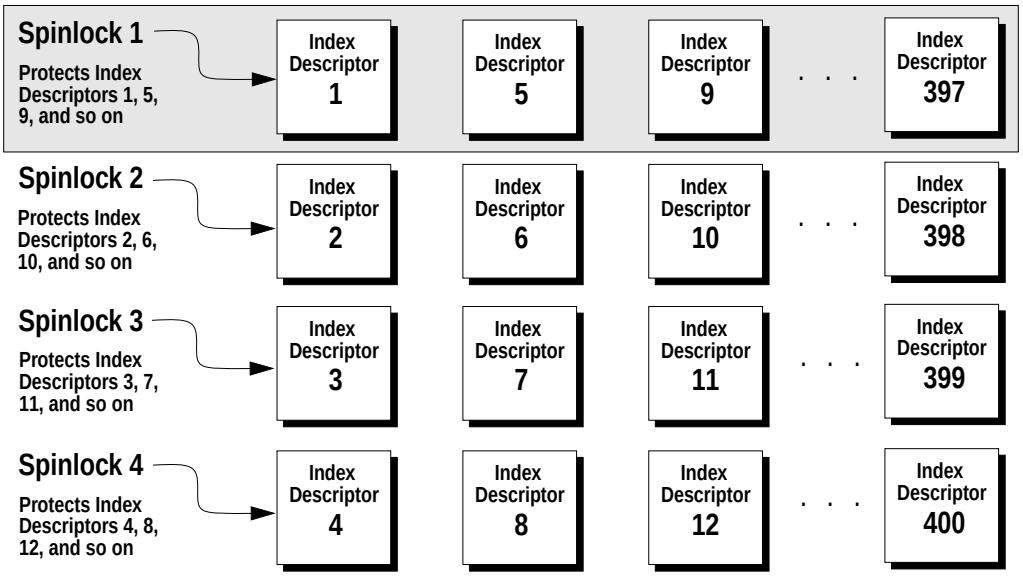


Figure 10-3: Relationship between spinlocks and index descriptors

Suppose there are 400 index metadata descriptors, or 400 rows in the index descriptors internal table. You have set the ratio to 100. This means that there will be 4 spinlocks in all: Spinlock 1 protects row 1;

Spinlock 2 protects row 2, Spinlock 3 protects row 3, and Spinlock 4 protects row 4. After that, Spinlock 1 protects the next available index descriptor, Index Descriptor 5, until every index descriptor is protected by a spinlock. This round-robin method of descriptor assignment reduces the chances of spinlock contention.

Sequential Assignment

Table lock spinlocks, configured by the **table lock spinlock ratio** parameter, use the sequential assignment method. The default configuration for table lock spinlock ratio is 20, which assign 20 rows in an internal hash table to each spinlock. The rows are divided up sequentially: the first spinlock protects the first 20 rows, the second spinlock protects the second 20 rows, and so on.

In theory, protecting one resource with one spinlock would provide the least contention for a spinlock and would result in the highest concurrency. In most cases, the default value for these spinlock ratios is probably best for your system. Change the ratio only if there is spinlock contention.

Use **sp_sysmon** to get a report on spinlock contention. See Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide* for information on spinlock contention.

Configuring Server Behavior

11 Setting Configuration Parameters

This chapter describes the Adaptive Server configuration parameters. A configuration parameter is a user-definable setting that you set with the system procedure **sp_configure**. Configuration parameters are used for a wide range of services, from basic to specific server operations, and for performance tuning.

Adaptive Server Configuration Parameters

The following table lists the Adaptive Server configuration parameters.

Configuration Parameters
additional network memory, page 11-89
address lock spinlock ratio, page 11-51
allow backward scans, page 11-96
allow nested triggers, page 11-97
allow procedure grouping, page 11-129
allow remote access, page 11-72
allow resource limits, page 11-97
allow sql server async i/o, page 11-36
allow resource limits, page 11-97
allow updates to system tables, page 11-98
auditing, page 11-129
audit queue size, page 11-130
cis bulk insert batch size, page 11-31
cis connect timeout, page 11-32
cis cursor rows, page 11-32
cis packet size, page 11-33
cis rpc handling, page 11-34
configuration file, page 11-47
cpu accounting flush interval, page 11-99
cpu grace time, page 11-100

Configuration Parameters (continued)
current audit table, page 11-131
deadlock checking period, page 11-52
deadlock retries, page 11-53
default character set id, page 11-48
default database size, page 11-101
default fill factor percent, page 11-102
default language id, page 11-48
default network packet size, page 11-72
default sortorder id, page 11-49
disable character set conversions, page 11-49
disk i/o structures, page 11-37
dump on conditions, page 11-103
enable cis, page 11-34
enable rep agent threads, page 11-95
esp execution priority, page 11-43
esp execution stacksize, page 11-44
esp unload dll, page 11-44
event buffers per engine, page 11-103
event log computer name (Windows NT only), page 11-40
event logging (Windows NT only), page 11-41
executable codesize + overhead, page 11-61
freelock transfer block size, page 11-55
global async prefetch limit, page 11-26
housekeeper free write percent, page 11-104
i/o accounting flush interval, page 11-108
i/o polling process count, page 11-109
identity burning set factor, page 11-106
identity grab size, page 11-107
lock promotion HWM, page 11-110
lock promotion LWM, page 11-111
lock promotion PCT, page 11-112

Configuration Parameters (continued)
lock shared memory, page 11-91
log audit logon failure, page 11-42
log audit logon success, page 11-42
max async i/os per engine, page 11-81
max async i/os per server, page 11-81
max engine freelocks, page 11-56
max cis remote connections, page 11-35
max cis remote servers, page 11-36
maximum dump conditions, page 11-113
max network packet size, page 11-74
max number network listeners, page 11-77
max online engines, page 11-94
max parallel degree, page 11-86
max roles enabled per user, page 11-132
max scan parallel degree, page 11-87
max SQL text monitored, page 11-91
memory alignment boundary, page 11-26
memory per worker process, page 11-88
number of alarms, page 11-113
number of aux scan descriptors, page 11-114
number of devices, page 11-38
number of index trips, page 11-27
number of languages in cache, page 11-50
number of large i/o buffers, page 11-21
number of locks, page 11-58
number of mailboxes, page 11-117
number of messages, page 11-118
number of oam trips, page 11-28
number of open databases, page 11-62
number of open indexes, page 11-64
number of open objects, page 11-66

Configuration Parameters (continued)
number of pre-allocated extents, page 11-118
number of remote connections, page 11-78
number of remote logins, page 11-78
number of remote sites, page 11-79
number of sort buffers, page 11-119
number of user connections, page 11-137
number of worker processes, page 11-86
open index hash spinlock ratio, page 11-69
open index spinlock ratio, page 11-70
open object spinlock ratio, page 11-71
o/s file descriptors, page 11-83
page lock spinlock ratio, page 11-59
page utilization percent, page 11-39
partition groups, page 11-120
partition spinlock ratio, page 11-121
permission cache entries, page 11-139
print deadlock information, page 11-122
print recovery information, page 11-22
procedure cache percent, page 11-29
recovery interval in minutes, page 11-22
remote server pre-read packets, page 11-79
runnable process search count, page 11-123
secure default login, page 11-133
select on syscomments.text column, page 11-134
shared memory starting address, page 11-84
size of auto identity column, page 11-124
SQL Perfmon Integration (Windows NT only), page 11-125
sql server clock tick length, page 11-126
stack guard size, page 11-140
stack size, page 11-143
start mail session (Windows NT only), page 11-45

Configuration Parameters (continued)
suspend audit when device full, page 11-134
systemwide password expiration, page 11-135
table lock spinlock ratio, page 11-60
tape retention in days, page 11-25
tcp no delay, page 11-80
time slice, page 11-127
total data cache size, page 11-30
total memory, page 11-92
unified login required (Windows NT Only), page 11-136
upgrade version, page 11-128
user log cache size, page 11-144
user log cache spinlock ratio, page 11-145
use security services (Windows NT Only), page 11-137
xp_cmdshell context, page 11-46

What Are Configuration Parameters?

Configuration parameters are user-definable settings that control various aspects of Adaptive Server's behavior. Adaptive Server supplies default values for all configuration parameters. You can use configuration parameters to tailor Adaptive Server for an installation's particular needs.

Read this chapter carefully to determine which configuration parameters you should reset to optimize server performance. Also, see the *Performance and Tuning Guide* for further information on using **sp_configure** to tune Adaptive Server.

◆ ***WARNING!***

Change configuration parameters with caution. Arbitrary changes in parameter values can adversely affect Adaptive Server performance and other aspects of server operation.

The Adaptive Server Configuration File

Adaptive Server stores the values of configuration parameters in a configuration file, which is an ASCII text file. When you install a new Adaptive Server, your parameters are set to the default configuration; the default name of the file is *server_name.cfg*, and the default location of the file is the Sybase installation directory ($SYBASE). When you change a configuration parameter, Adaptive Server saves a copy of the old configuration file as *server_name.001*, *server_name.002*, and so on. Adaptive Server writes the new values to the file *server_name.cfg* or to a file name you specify at start-up.

How to Modify Configuration Parameters

You set or change configuration parameters in one of the following ways:

- By executing the system procedure **sp_configure** with the appropriate parameters and values,
- By hand-editing your configuration file and then invoking **sp_configure** with the **configuration file** option, or
- By specifying the name of a configuration file at start-up.

Configuration parameters are either **dynamic** or **static**. Dynamic parameters go into effect as soon as you execute **sp_configure**. Static parameters require Adaptive Server to reallocate memory, so they take effect only after Adaptive Server has been restarted. The description of each parameter indicates whether it is static or dynamic. Adaptive Server writes the new value to the system table *sysconfigures* and to the configuration file when you change the value, not when you restart Adaptive Server. The current configuration file and *sysconfigures* reflect configured values, not run values. The system table *syscurconfigs* reflects current run values of configuration parameters.

Who Can Modify Configuration Parameters

The roles required for using **sp_configure** are as follows:

- Any user can execute **sp_configure** to display information about parameters and their current values.
- Only a System Administrator and System Security Officer can execute **sp_configure** to modify configuration parameters.

- Only a System Security Officer can execute **sp_configure** to modify values for

  - **allow procedure grouping**
  - **allow updates to system tables**
  - **auditing**
  - **audit queue size**
  - **current audit table**
  - **max roles enabled per user**
  - **allow remote access**
  - **secure default login**
  - **select on syscomments.text column**
  - **suspend audit when device full**
  - **systemwide password expiration**
  - **unified login required (Windows NT Only)**
  - **use security services (Windows NT Only)**

Getting Help Information on Configuration Parameters

Use either **sp_helpconfig** or **sp_configure** to get help information on a particular configuration parameter. For example:

```
sp_helpconfig "number of open"

Configuration option is not unique.

option_name                    config_value run_value
------------------------------ ------------ -----------
number of open databases                 12          12
number of open indexes                  500         500
number of open objects                  500         500
```

```
sp_helpconfig "number of open indexes"

number of open indexes sets the maximum number of indexes that can be
open at one time on SQL Server. The default value is 500.

Minimum Value Maximum Value Default Value Current Value Memory Used

------------- ------------- ------------- ------------- -----------

          100    2147483647           500           500         208
```

```
sp_configure "number of open indexes"

Parameter Name           Default  Memory Used  Config Value  Run Value
-----------------------  -------  -----------  ------------  ---------
number of open indexes       500          208           500        500
```

For more information on **sp_helpconfig**, see “Using sp_helpconfig to Get Help on Configuration Parameters” on page 8-6.

Using *sp_configure*

The system procedure **sp_configure** displays and resets configuration parameters. You can restrict the number of parameters displayed by **sp_configure** using **sp_displaylevel** to set your display level to one of three values:

- Basic
- Intermediate
- Comprehensive

For information about display levels, see "User-Defined Subsets of the Parameter Hierarchy: Display Levels" on page 11-16. For information about **sp_displaylevel**, see the *Adaptive Server Reference Manual*.

Each parameter belongs to a group based on the area of server behavior it affects. See "The Parameter Hierarchy" on page 11-15 for more information.

Table 11-1 describes the syntax for **sp_configure**. The information in the "Effect" column assumes that your display level is set to "comprehensive."

Table 11-1: sp_configure syntax

Command	Effect
sp_configure	Displays all configuration parameters by group, their current values, their default values, the value to which they have most recently been set, and the amount of memory used by this particular setting.
sp_configure "*parameter***"**	Displays current value, default value, most recently changed value, and amount of memory used by setting for all parameters matching parameter.
sp_configure "*parameter***",** *value*	Resets *parameter* to *value*.
sp_configure "*parameter***", *0*, "default"**	Resets parameter to its default value.
sp_configure "*group_name***"**	Displays all configuration parameters in *group_name*, their current values, their default values, the values to which they were recently been set, and the amount of memory used by each setting.

Table 11-1: sp_configure syntax (continued)

Command	Effect
sp_configure "**configuration file**", 0, "*sub_command*", "*file_name*"	Sets configuration parameters from the configuration file. See "Using sp_configure with a Configuration File" on page 11-10 for descriptions of the parameters.

Syntax Elements

In Table 11-1 the following variables are used:

- *parameter* – is any valid Adaptive Server configuration parameter or parameter substring.
- *value* – is any integer within the valid range for that parameter. (See the descriptions of the individual parameters for valid range information.) Parameters that are toggles have only two valid values: 1 (on) and 0 (off).
- *group_name* – is the name of any group in the parameter hierarchy.

Parameter Parsing

sp_configure parses each parameter (and parameter name fragment) as "*%parameter%*". A string that does not uniquely identify a particular parameter returns values for all parameters matching the string. For example:

```
sp_configure "lock"
```

returns values for all configuration parameters that include "lock", such as **lock shared memory**, **number of locks**, **lock promotion HWM**, **server clock tick length**, **print deadlock information**, and **deadlock retries**.

➤ ***Note***

If you attempt to set a parameter value with a nonunique parameter name fragment, **sp_configure** returns the current values for all parameters matching the fragment and asks for a unique parameter name.

Using *sp_configure* with a Configuration File

Adaptive Server configuration can be done either interactively, by using **sp_configure** as described above, or noninteractively, by instructing Adaptive Server to read values from an edited or restored version of the configuration file.

Configuration files can be used for several reasons:

- You can replicate a specific configuration across multiple servers by using the same configuration file.
- You can use a configuration file as a baseline for testing configuration values on your server.
- You can use a configuration file to do validation checking on parameter values before actually setting the values.
- You can create multiple configuration files and switch between them as your resource needs change.

You can make a copy of the configuration file using the system procedure **sp_configure** with the parameter "**configuration file**" and then edit it by hand at the operating system level. Then, you can use **sp_configure** (with the parameter "**configuration file**") to instruct Adaptive Server to read values from the edited file. Or you can specify the name of the configuration file at start-up.

For information on editing the file by hand, see "Editing the Configuration File by Hand" on page 11-12. For information on specifying the name of the configuration file at start-up, see "Starting Adaptive Server with a Configuration File" on page 11-14.

Naming Tips for the Configuration File

Each time you modify a configuration parameter with **sp_configure**, Adaptive Server creates copies of the outdated configuration file, using the naming convention *server_name.001*, *server_name.002*, *server_name.003*...*server_name.999*.

If you want to work with a configuration file with a name other than the default name, and you keep the *server_name* part of the file name, be sure to include at least one alphabetic character in the extension. Alternatively, you can change the *server_name* part of the file name. Doing this avoids confusion with the backup configuration files generated by Adaptive Server when you modify a parameter.

Using *sp_configure* to Read or Write the Configuration File

The syntax for using the **configuration file** option with **sp_configure** is:

```
sp_configure "configuration file", 0, "subcommand",
   "file_name"
```

where:

- "configuration file" (include quotes) specifies the configuration file parameter.
- 0 must be included as the second parameter to **sp_configure** for backward compatibility.
- "*subcommand*" is one of the commands described below.
- *file_name* specifies the configuration file you want to use in conjunction with any *subcommand*. If you do not specify a directory as part of the file name, the directory where Adaptive Server was started is used.

Parameters for Using Configuration Files

The four parameters described below can be used with configuration files.

write

write creates *file_name* from the current configuration. If *file_name* already exists, a message is written to the error log; the existing file is renamed using the convention *file_name.001*, *file_name.002*, and so on. If you have changed a static parameter, but you have not restarted your server, **write** gives you the **currently running value** for that parameter. If you do not specify a directory with *file_name*, the file is written to the directory from which Adaptive Server was started.

read

read performs validation checking on values contained in *file_name* and reads those values that pass validation into the server. If any parameters are missing from *file_name*, the current values for those parameters are used.

If the value of a static parameter in *file_name* is different from its current running value, **read** fails and a message is printed. However, validation is still performed on the values in *file_name*.

verify

verify performs validation checking on the values in *file_name*. This is useful if you have edited the configuration file by hand, as it prevents you from attempting to configure your server with invalid configuration values.

restore

restore creates *file_name* with the most recently configured values. If you have configured static parameters to new values, this subcommand will write the configured, not the currently running, values to the file. This is useful if all copies of the configuration file have been lost and you need to generate a new copy. If you do not specify a directory with *file_name*, the file is written to the directory from which Adaptive Server was started.

Examples

```
sp_configure "configuration file", 0, "read",
"srv.config"
```

The above example performs validation checking on the values in the file *srv.config* and reads the parameters that pass validation into the server. Current run values are substituted for values that do not pass validation checking.

```
sp_configure "configuration file", 0, "write",
"my_server.config"
```

The above example creates the file *my_server.config* and writes the current configuration values the server is using to that file.

```
sp_configure "configuration file", 0, "verify",
"generic.config"
```

The above example runs validation checking on the values in the file *generic.config*.

```
sp_configure "configuration file", 0, "restore",
"restore.config"
```

The above example writes configured values to the file *restore.config*.

Editing the Configuration File by Hand

The configuration file is an operating system ASCII file that can be edited with any text editor that can save files in ASCII format. The syntax for each parameter is:

***parameter_name*={*value* | DEFAULT}**

where *parameter_name* is the name of the parameter you want to specify, *value* is the numeric value for set *parameter_name*, and "DEFAULT" specifies that you want to use the default value for *parameter_name*.

Examples:

```
deadlock retries = 1
```

specifies the that transaction can retry to acquiring a lock one time when deadlocking occurs during an index page split or shrink.

```
cpu accounting flush interval=DEFAULT
```

specifies that the default value for the parameter **cpu accounting flush interval** should be used.

When you hand-edit a configuration file, your edits are not validated until you check the file using the **verify** option, read the file with the **read** option, or restart Adaptive Server with that configuration file.

If all your configuration files are lost or corrupted, you can re-create one from a running server by using the **restore** subcommand and specifying a name for the new file. The parameters in the new file will be set to the values with which your server is currently running.

Permissions for Configuration Files

Configuration files are nonencrypted ASCII text files. By default, they are created with read and write permissions set for the file owner and read permission set for all other users. (If you created the configuration file at the operating system level, you are the file owner; if you created the configuration file from Adaptive Server, using the **write** or **restore** parameter, the file owner is the user who started Adaptive Server. Usually, this is the user "sybase." To restrict access to configuration files, use your operating system's file permission command to set read, write, and execute permissions as appropriate.

➤ ***Note***

You need to set permissions accordingly on **each** configuration file created.

Backing Up Configuration Files

Configuration files are not automatically backed up when you back up the *master* database. They are operating system files, and you should back them up in the same way you back up your other operating system files.

Checking the Name of the Configuration File Currently in Use

The output from **sp_configure** truncates the name of the configuration file due to space limitations. To see the full name of the configuration file, use:

```
select s1.value2
from syscurconfigs s1, sysconfigures s2
where s1.config = s2.config
and s2.name = "configuration file"
```

Starting Adaptive Server with a Configuration File

By default, Adaptive Server reads the configuration file *server_name.cfg* in the start-up directory when it starts. If this file does not exist, it creates a new file and uses Adaptive Server defaults for all values.

You can start Adaptive Server with a specified configuration file. For more information, see the *Utility Programs* manual for your platform.

If the configuration file you specify does not exist, Adaptive Server prints an error message and does not start.

If the command is successful, the file *server_name.bak* is created. This file contains the configuration values stored in *sysconfigures* prior to the time *sysconfigures* was updated with the values read in from the configuration file you specified. This file is overwritten with each subsequent start-up.

Configuration File Errors

When there are errors in the configuration file, Adaptive Server may not start or may use default values.

Adaptive Server uses default values:

- If there are illegal values. For example, if a parameter requires a numeric value, and the configuration file contains a character string, Adaptive Server uses the default value.
- If values are below the minimum allowable value.

The Parameter Hierarchy

Configuration parameters are grouped according to the area of Adaptive Server behavior they affect. This makes it easier to identify all parameters that might need to be tuned in order to improve a particular area of Adaptive Server performance.

The groups are:

- Backup and Recovery
- Cache Manager
- Component Integration Services Administration
- Disk I/O
- Error Log
- Extended Stored Procedures
- General Information
- Languages
- Lock Manager
- Memory Use
- Metadata Caches
- Network Communication
- O/S Resources
- Parallel Queries
- Physical Memory
- Processors
- Rep Agent Thread Administration
- SQL Server Administration
- Security Related
- User Environment

Although each parameter has a primary group to which it belongs, many have secondary groups to which they also belong. For instance, the parameter **number of remote connections** belongs primarily to the Network Communication group, but it also belongs secondarily to the Adaptive Server Administration group and the Memory Use group. This reflects the fact that some parameters have implications for a number of areas of Adaptive Server behavior. **sp_configure** displays parameters in all groups to which they belong.

The syntax for displaying all groups and their associated parameters, and the current values for the parameters, is:

```
sp_configure
```

➤ ***Note***

The number of parameters **sp_configure** returns depends on the value to which you have your display level set. If your display level is set to "basic," **sp_configure** does not return parameters that are designated as either "intermediate" or "comprehensive." If your display level is set to "intermediate," **sp_configure** returns the parameters designated as "basic" and "intermediate," but not those designated as "comprehensive." See "User-Defined Subsets of the Parameter Hierarchy: Display Levels" on page 11-16 for further information about display levels.

The syntax for displaying a particular group and its associated parameter is:

```
sp_configure "group_name"
```

where *group_name* is the name of the group you are interested in. For example, to display the Disk I/O group, type:

```
sp_configure "Disk I/O"
```

```
Group: Disk I/O

Parameter Name           Default Memory Used Config Value Run Value
--------------           ------- ----------- ------------  ---------
allow sql server async i/o     1           0             1         1
disk i/o structures          256           0           256       256
number of devices             10           0            10        10
page utilization percent      95           0            95        95
```

➤ ***Note***

If the server uses a case-insensitive sort order, **sp_configure** with no parameters returns a list of all configuration paramemters and groups in alphabetical order with no grouping displayed.

User-Defined Subsets of the Parameter Hierarchy: Display Levels

Depending on your use of Adaptive Server, you may need to adjust some parameters more frequently than others. You may find it is easier to work with a subset of parameters than having to see the

entire group when you are working with only a few. You can set your display level to one of three values to give you the subset of parameters that best suits your working style.

The default display level is "comprehensive." When you set your display level, the setting persists across multiple sessions. However, you can reset it at any time to see more or fewer configuration parameters.

- "Basic" level shows just the most basic parameters. It is appropriate for very general server tuning.
- "Intermediate" level shows you parameters that are somewhat more complex, as well as showing you all the "basic" parameters. This level is appropriate for a moderately complex level of server tuning.
- "Comprehensive" level shows you all the parameters, including the most complex ones. This level is appropriate for users doing highly detailed server tuning.

The syntax for showing your current display level is:

```
sp_displaylevel
```

The syntax for setting your display level is:

```
sp_displaylevel user_name [, basic | intermediate |
   comprehensive]
```

where *user_name* is your Adaptive Server login name.

The Effect of the Display Level on *sp_configure* Output

If your display level is set to either "basic" or "intermediate," **sp_configure** returns only a subset of the parameters that are returned when your display level is set to "comprehensive." For instance, if your display level is set to "intermediate," and you want to see the parameters in the Languages group, type:

```
sp_configure "Languages"
```

The output would look like this:

```
Group: Languages
Parameter Name               Default Memory Used Config Value Run Value
---------------------------- ------- ----------- ------------ ---------
default character set id           1           0            1         1
default language id                0           0            0         0
number of languages in cache       3           4            3         3
```

However, this is only a subset of the parameters in the Languages group, because some parameters in that group are displayed only at the "comprehensive" level.

The *reconfigure* Command

Pre-11.0 SQL Server releases required you to execute **reconfigure** after executing **sp_configure**. Beginning with SQL Server release 11.0, this was longer required. The **reconfigure** command still exists, but it does not have any effect. It is included in Adaptive Server release 11.5 so that pre-11.0 SQL scripts can run without modification.

Scripts using **reconfigure** will still run in the current release, but you should change them at your earliest convenience because **reconfigure** will not be supported in future releases of Adaptive Server.

Performance Tuning with *sp_configure* and *sp_sysmon*

sp_sysmon is a system procedure that monitors Adaptive Server performance and generates statistical information that describes the behavior of your Adaptive Server system. See Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide* for information about how to use **sp_sysmon**.

You can run **sp_sysmon** before and after using **sp_configure** to adjust configuration parameters. The output gives you a basis for performance tuning and lets you observe the results of configuration changes.

This chapter includes cross-references to the *Performance and Tuning Guide* for the **sp_configure** parameters that can affect Adaptive Server performance.

Output from *sp_configure*

The sample output below shows the kind of information **sp_configure** prints if you have your display level set to "comprehensive" and you execute it with no parameters. The values it prints will vary, depending on your platform and on what values you have already changed.

```
sp_configure
```

```
Group: General Information
Parameter Name       Default Memory Used Config Value Run Value
--------------       ------- ----------- ------------ ---------
configuration file         0           0            0 /remote/pub

Group: Backup/Recovery

Parameter Name         Default Memory Used Config Value Run Value
--------------         ------- ----------- ------------ ---------
recovery interval in minutes 5           0            5         5
tape retention in days       0           0            0         0
recovery flags               0           0           0        0

...
```

➤ ***Note***

All configuration groups and parameters will appear in output if your display level is set to “comprehensive”.

The “Default” column displays the value Adaptive Server is shipped with. If you do not explicitly reconfigure a parameter, it retains its default value.

The “Memory Used” column displays the amount of memory used (in kilobytes) by the parameter at its current value. Some related parameters draw from the same memory pool. For instance, the memory used for **stack size** and **stack guard size** is already accounted for in the memory used for **number of user connections**. If you added the memory used by each of these parameters separately, it would total more than the amount actually used. In the “Memory Used” column, parameters that “share” memory with other parameters are marked with a hash mark (“#”).

The “Config Value” column displays the most recent value to which the configuration parameter has been set with **sp_configure**. When you execute **sp_configure** to modify a dynamic parameter:

- The configuration and run values are updated
- The configuration file is updated
- The change takes effect immediately

When you modify a static parameter:

- The configuration value is updated
- The configuration file is updated
- The change takes effect only when you restart Adaptive Server

The "Run Value" column displays the value Adaptive Server is currently using. It changes when you modify a dynamic parameter's value with **sp_configure** and, for static parameters, after you restart Adaptive Server.

The *sysconfigures* and *syscurconfigs* Tables

The report displayed by **sp_configure** is constructed mainly from the *master..sysconfigures* and *master..syscurconfigs* system tables, with additional information coming from *sysattributes*, *sysdevices*, and other system tables.

The *value* column in the *sysconfigures* table records the last value set from **sp_configure** or the configuration file; the *value* column in *syscurconfigs* stores the value currently in use. For dynamic parameters, the two values match; for static parameters, which require a restart of the server to take effect, the two values are different if the values have been changed since Adaptive Server was last started. The values may also be different when the default values are used. In this case, *sysconfigures* stores 0, and *syscurconfigs* stores the value that Adaptive Server computes and uses.

sp_configure performs a join on *sysconfigures* and *syscurconfigs* to display the values reported by **sp_configure**.

Querying *syscurconfigs* and *sysconfigures*: An Example

You might want to query *sysconfigures* and *syscurconfigs* to get information organized the way you want. For example, **sp_configure** without any arguments lists the memory used for configuration parameters, but it does not list minimum and maximum values. You can query these system tables to get a complete list of current memory usage, as well as minimum, maximum, and default values, with the following query:

```
select b.name, memory_used, minimum_value,
maximum_value, defvalue
from master.dbo.sysconfigures b,
master.dbo.syscurconfigs c
where b.config *= c.config and parent != 19
and b.config > 100
```

Details on Configuration Parameters

The following sections give both summary and detailed information about each of the configuration parameters. Parameters are listed by group; within each group, they are listed alphabetically.

In many cases, the maximum allowable values for configuration parameters are extremely high. The maximum value for your server is usually limited by available memory, rather than by **sp_configure** limitations.

Backup and Recovery

The following parameters configure Adaptive Server for backing up and recovering data:

number of large i/o buffers

Summary Information	
Name in pre-11.0 release	N/A
Default value	6
Valid values	1–32
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of large i/o buffers** parameter sets the number of 16K buffers reserved for performing large I/O for certain Adaptive Server utilities. These large I/O buffers are used primarily by the **load database** command. Each **load database** command uses one buffer, regardless of the number of stripes specified in the **load database** command. These buffers are not used by **load transaction**. If you need to perform more than six **load database** commands concurrently, configure one large I/O buffer for each **load database** command.

create database and **alter database** use these buffers for large I/O while clearing database pages. Each instance of **create database** or **load database** can use up to 8 large I/O buffers.

These buffers are also used by disk mirroring and by some **dbcc** commands.

print recovery information

Summary Information	
Name in pre-11.0 release	**recovery flags**
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Intermediate
Required role	System Administrator

The **print recovery information** parameter determines what information Adaptive Server displays on the console during recovery. (Recovery is done on each database at Adaptive Server start-up and when a database dump is loaded.) The default value is 0, which means that Adaptive Server displays only the database name and a message saying that recovery is in progress. The other value is 1, which means that Adaptive Server displays information about each individual transaction processed during recovery, including whether it was aborted or committed.

recovery interval in minutes

Summary Information	
Name in pre-11.0 release	**recovery interval**
Default value	5
Range of values	1–32767
Status	Dynamic
Display level	Basic
Required role	System Administrator

The **recovery interval in minutes** parameter sets the maximum number of minutes per database that Adaptive Server uses to complete its recovery procedures in case of a system failure. The recovery procedure rolls transactions backward or forward, starting from the transaction that the checkpoint process indicates as the oldest active

transaction. The recovery process has more or less work to do depending on the value of **recovery interval in minutes**.

Adaptive Server estimates that 6000 rows in the transaction log require 1 minute of recovery time. However, different types of log records can take more or less time to recover. If you set **recovery interval in minutes** to 3, the checkpoint process writes changed pages to disk only when *syslogs* contains more than 18,000 rows since the last checkpoint.

➤ ***Note***

The recovery interval has no effect on long-running, minimally logged transactions (such as **create index**) that are active at the time Adaptive Server fails. It may take as much time to reverse these transactions as it took to run them. To avoid lengthy delays, dump each database after index maintenance operations.

Adaptive Server uses the **recovery interval in minutes** setting and the amount of activity on each database to decide when to checkpoint each database. When Adaptive Server checkpoints a database, it writes all **dirty pages** (data pages in cache that have been modified) to disk. This may create a brief period of high I/O, called a **checkpoint spike**.The checkpoint also performs a few other maintenance tasks, including truncating the transaction log for each database for which the **truncate log on chkpt** option has been set. About once per minute, the sleeping checkpoint process "wakes up," checks the **truncate log on chkpt** setting, and checks the recovery interval to determine if a checkpoint is needed. Figure 11-1 shows the logic used by Adaptive Server during this process.

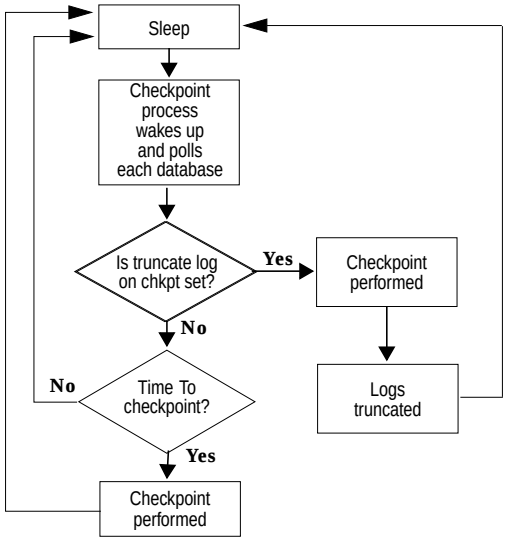


Figure 11-1: The checkpoint process

You may want to change the recovery interval if your application and its use change. For example, you may want to shorten the recovery interval when there is an increase in update activity on Adaptive Server. Shortening the recovery interval causes more frequent checkpoints, with smaller, more frequent checkpoint spikes, slows the system slightly. On the other hand, setting the recovery interval too high might cause the recovery time to be unacceptably long. (The spikes caused by checkpointing can be reduced by reconfiguring the **housekeeper free write percent** parameter. See "housekeeper free write percent" on page 11-104 for further information.) For more information on the performance implications of **recovery interval in minutes**, see "Speed of Recovery" on page 16-39 in the *Performance and Tuning Guide*.

Use **sp_sysmon** to determine how a particular recovery interval affects the system. See "Recovery Management" on page 24-85 in the *Performance and Tuning Guide* for more information.

tape retention in days

Summary Information	
Name in pre-11.0 release	**tape retention**
Default value	0
Range of values	0–365
Status	Static
Display level	Intermediate
Required role	System Administrator

The **tape retention in days** parameter specifies the number of days you intend to retain each tape after it has been used for either a database or a transaction log dump. It is intended to keep you from accidentally overwriting a dump tape.

For example, if you have set **tape retention in days** to 7 days, and you try to use the tape before 7 days have elapsed since the last time you dumped to that tape, Backup Server issues a warning message to the tape operator.

You can override the warning by using the **with init** option when executing the dump command. Be aware that doing this will cause the tape to be overwritten and all data on the tape to be lost.

Both the **dump database** and **dump transaction** commands provide a **retaindays** option, which overrides the **tape retention in days** value for a particular dump. See "Protecting Dump Files from Being Overwritten" on page 21-22 for more information.

Cache Manager

The parameters in this group configure the data and procedure caches.

global async prefetch limit

Summary Information	
Name in pre-11.0 release	N / A
Default value	10
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **global async prefetch limit** parameter specifies the percentage of a buffer pool that can hold the pages brought in by asynchronous prefetch that have not yet been read. This parameter sets the limit for all pools in all caches for which the limit has not been set explicitly with the **sp_poolconfig** system procedure.

If the limit for a pool is exceeded, asynchronous prefetch is temporarily disabled until the percentage of unread pages falls below the limit.

memory alignment boundary

Summary Information	
Name in pre-11.0 release	**calignment**
Default value	2048
Range of values	2048–16384
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **memory alignment boundary** parameter determines the memory address boundary on which data caches are aligned.

Some machines perform I/O more efficiently when structures are aligned on a particular memory address boundary. In order to preserve this alignment, values for **memory alignment boundary** should always be multiples of 2K.

➤ ***Note***

The **memory alignment boundary** parameter is included for support of certain hardware platforms. Do not modify it unless you are instructed to do so by Sybase Technical Support.

number of index trips

Summary Information	
Name in pre-11.0 release	**cindextrips**
Default value	0
Range of values	0–65535
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **number of index trips** parameter specifies the number of times an aged index page traverses the most recently used / least recently used (MRU / LRU) chain before it is considered for swapping out. As you increase the value of **number of index trips**, index pages stay in cache for longer periods of time.

A data cache is implemented as an MRU / LRU chain. As the user threads access data and index pages, these pages are placed on the MRU end of the cache's MRU / LRU chain. In some high transaction environments (and in some benchmarks), it is desirable to keep index pages in cache, since they will probably be needed again soon. Setting **number of index trips** higher keeps index pages in cache longer; setting it lower allows index pages to be swapped out of cache sooner.

You do not need to set the number of index pages parameter for relaxed LRU pages. For more information on relaxed LRU pages, see Chapter 9, "Configuring Data Caches."

➤ ***Note***

If the cache used by an index is relatively small (especially if it shares space with other objects) and you have a high transaction volume, you should be careful not to set **number of index trips** too high. The cache can flood with pages that do not age out, and this may lead to the timing out of processes that are waiting for cache space.

number of oam trips

Summary Information	
Name in pre-11.0 release	**coamtrips**
Default value	0
Range of values	0–65535
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **number of oam trips** parameter specifies the number of times an **Object Allocation Map** (**OAM**) page traverses the MRU / LRU chain before it is considered for swapping out. The higher the value of **number of oam trips**, the longer aged OAM pages stay in cache.

Each table, and each index on a table, has an OAM page. The OAM page holds information on pages allocated to the table or index and is checked when a new page is needed for the index or table. (See "page utilization percent" on page 11-39 for further information.) A single OAM page can hold allocation mapping for between 2,000 and 63,750 data or index pages.

The OAM pages point to the allocation page for each allocation unit where the object uses space. The allocation pages, in turn, track the information about extent and page usage within the allocation unit.

In some environments and benchmarks that involve significant allocations of space (that is, massive bulk copy operations), keeping OAM pages in cache longer improves performance. Setting **number of oam trips** higher keeps OAM pages in cache.

➤ ***Note***

If the cache is relatively small and used by a large number of objects, you should not set **number of oam trips** too high. This may result in the cache being flooded with OAM pages that do not age out, and user threads may begin to time out.

procedure cache percent

Summary Information	
Name in pre-11.0 release	**procedure cache**
Default value	20
Range of values	1–99
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **procedure cache percent** parameter specifies the percentage of memory allocated to the procedure cache after Adaptive Server's memory needs are met. Adaptive Server's memory needs are the sum of memory necessary for locks, user connections, the code itself, which varies slightly from release to release, and other resources. The remaining memory is divided between the procedure cache and the data cache, according to the value to which **procedure cache percent** has been set.

Adaptive Server uses the procedure cache while running stored procedures. If the server finds a copy of a procedure already in the cache, it does not need to read it from the disk. Adaptive Server also uses space in the procedure cache to compile queries while creating stored procedures.

Since the optimum value for **procedure cache percent** is different from application to application, resetting it may improve Adaptive Server's performance. For example, if you run many different procedures or ad hoc queries, your application will use the procedure cache more heavily, so you may want to increase this value.

Many applications are tested during development with various procedures and ad hoc queries. You may want to try setting this

parameter to 50 during your development cycle and resetting it to 20 when your application becomes stable. For further information on configuring procedure caches, see “The Procedure Cache” on page 16-4 of the *Performance and Tuning Guide*.

total data cache size

Summary Information	
Name in pre-11.0 release	N / A
Default value	N / A
Range of values	N / A
Status	Calculated
Display level	Basic
Required role	System Administrator

The **total data cache size** parameter reports the amount of memory, in kilobytes, that is currently available for data, index, and log pages. It is a calculated value that is not directly user-configurable.

The amount of memory available for the data cache can be affected by a number of factors, including:

- The amount of physical memory available on your machine
- The values to which the following parameters are set:
  - **total memory**
  - **number of user connections**
  - **total procedure cache percent**
  - **number of open databases**
  - **number of open objects**
  - **number of open indexes**
  - **number of devices**

A number of other parameters also affect the amount of available memory, but to a lesser extent.

For information on how Adaptive Server allocates memory and for information on data caches, see “Details on Configuration Parameters” on page 11-21.

Component Integration Services Administration

The following parameters configure Adaptive Server for Component Integration Services.

cis bulk insert batch size

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cis bulk insert batch size** parameter determines how many rows from the source table(s) are to be bulk copied into the target table as a single batch using **select into**.

If left at zero (the default), all rows are copied as a single batch. Otherwise, after the count of rows specified by this parameter has been copied to the target table, the server issues a bulk commit to the target server, causing the batch to be committed.

If a normal client-generated bulk copy operation (such as that produced by the **bcp** utility) is received, then the client is expected to control the size of the bulk batch, and the server ignores the value of this configuration parameter.

cis connect timeout

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–32767
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cis connect timeout** parameter determines the wait time in seconds for a successful Client-Library connection. By default no timeout is provided.

cis cursor rows

Summary Information	
Name in pre-11.0 release	N / A
Default value	50
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cis cursor rows** parameter specifies the cursor row count for **cursor open** and **cursor fetch** operations. Increasing this value means more rows will be fetched in one operation. This increases speed but requires more memory. The default is 50.

cis packet size

Summary Information	
Name in pre-11.0 release	N / A
Default value	512
Range of values	512–32768
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cis packet size** parameter specifies the size of Tabular Data Stream™ (TDS) packets that are exchanged between the server and a remote server when connection is initiated.

The default packet size on most systems is 512 bytes, and this may be adequate for most applications. However, larger packet sizes may result in significantly improved query performance, especially when *text* and *image* or bulk data is involved.

If a packet size larger than the default is specified, and the requested server is a System 10 or later Adaptive Server, then the target server must be configured to allow variable-length packet sizes. Adaptive Server configuration parameters of interest in this case are:

- **additional netmem**
- **maximum network packet size**

cis rpc handling

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cis rpc handling** parameter specifies the default method for remote procedural call (RPC) handling. Setting **cis rpc handling** to 0 sets the Adaptive Server site handler as the default RPC handling mechanism. Setting the parameter to 1 forces RPC handling to use Component Integration Services access methods. For more information, see the discussion on **set cis rpc handling** in the *Component Integration Services User's Guide*.

enable cis

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **enable cis** parameter enables or disables Component Integration Services.

max cis remote connections

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **max cis remote connections** parameter specifies the maximum number of concurrent Client-Library connections that can be made to remote servers by Component Integration Services.

By default, Component Integration Services allows up to four connections per user to be made simultaneously to remote servers. If you set the maximum number of users to 25, up to 100 simultaneous Client-Library connections would be allowed by Component Integration Services.

If this number does not meet the needs of your installation, you can override the setting by specifying exactly how many outgoing Client-Library connections you want the server to be able to make at one time.

max cis remote servers

Summary Information	
Name in pre-11.0 release	N / A
Default value	25
Range of values	10–32767
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **max cis remote servers** parameter specifies the number of concurrent servers that can be accessed from within the server using Client-Library connections.

Disk I/O

The parameters in this group configure Adaptive Server's disk I / O.

allow sql server async i/o

Summary Information	
Name in pre-11.0 release	**T1603** (trace flag)
Default value	1
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **allow sql server async i/o** parameter enables Adaptive Server to run with asynchronous disk I / O. In order to use asynchronous disk I / O, you have to enable it on **both** Adaptive Server **and** your operating system. See your operating system documentation for information on enabling asynchronous I / O at the operating system level.

In all circumstances, disk I / O runs faster asynchronously than synchronously. This is because when Adaptive Server issues an

asynchronous I/O, it does not have to wait for a response before issuing further I/Os.

disk i/o structures

Summary Information	
Name in pre-11.0 release	**cnblkio**
Default value	256
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **disk i/o structures** parameter specifies the initial number of disk I/O control blocks Adaptive Server allocates at start-up.

User processes require a disk I/O control block before Adaptive Server can initiate an I/O request for the process. The memory for disk I/O control blocks is preallocated when Adaptive Server starts. You should configure **disk i/o structures** to as high a value as your operating system allows, to minimize the chance of running out of disk I/O structures. Refer to your operating system documentation for information on concurrent disk I/Os.

Use **sp_sysmon** to determine whether you need to allocate more disk I/O structures. See "Disk I/O Structures" on page 24-91 in the *Performance and Tuning Guide*. You can set the **max asynch i/os per server** configuration parameter to the same value as **disk i/o structures**. See "max async i/os per server" on page 11-81 for more information.

number of devices

Summary Information	
Name in pre-11.0 release	**devices**
Default value	10
Range of values	1–256
Status	Static
Display level	Basic
Required role	System Administrator

The **number of devices** parameter controls the number of database devices Adaptive Server can use. It does not include devices used for database or transaction log dumps; it only includes devices used by Adaptive Server to store databases.

When you execute the **disk init** command, you assign the device number (the *vdevno*). Each device number must be unique among the device numbers used by Adaptive Server. The number 0 is reserved for the master device. Legal numbers are 1–256. However, the highest number must be 1 less than the number of database devices you have configured for Adaptive Server. For example, if you configured your server for 10 devices, the legal range of device numbers is 1–9.

To determine which numbers are currently in use, run **sp_helpdevice** and look in the *device_number* column of output.

If you drop a device with **sp_dropdevice**, you cannot reuse its *vdevno* until you restart Adaptive Server.

If you want to lower the **number of devices** value after you have added database devices, you must first check to see what devices numbers are already in use by database devices. The following command prints the highest value in use:

```
select max(low/power(2,24))+1
    from master..sysdevices
```

◆ ***WARNING!***

If you set the number of devices value too low in your configuration file, Adaptive Server cannot start. You can find the devices in use by checking the *sysdevices* system table.

page utilization percent

Summary Information	
Name in pre-11.0 release	N/A
Default value	95
Range of values	1–100
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **page utilization percent** parameter is used during page allocations to control whether Adaptive Server scans a table's OAM (**Object Allocation Map**) to find unused pages or simply allocates a new extent to the table. (See "number of oam trips" on page 11-28 for more information on the OAM.) The **page utilization percent** parameter is a performance optimization for servers with very large tables; it reduces the time needed to add new space.

If **page utilization percent** is set to 100, Adaptive Server scans through all OAM pages to find unused pages allocated to the object before allocating a new extent. When this parameter is set lower than 100, Adaptive Server compares the **page utilization percent** setting to the ratio of used and unused pages allocated to the table, as follows:

```
100 * used pages/(used pages + unused pages)
```

If the **page utilization percent** setting is lower than the ratio, Adaptive Server allocates a new extent instead of searching for the unused pages.

For example, when inserting data into a 10GB table that has 120 OAM pages and only 1 unused data page:

- A **page utilization percent** of 100 tells Adaptive Server to scan through all 120 OAM pages to locate an unused data page.
- A **page utilization percent** of 95 allows Adaptive Server to allocate a new extent to the object, because 95 is lower than the ratio of used pages to used and unused pages.

A low **page utilization percent** value results in more unused pages. A high **page utilization percent** value slows page allocations in very large tables, as Adaptive Server performs an OAM scan to locate each unused page before allocating a new extent. This increases logical and physical I/O.

If page allocations (especially in the case of large inserts) seem to be slow, you can lower the value of **page utilization percent**, but be sure to reset it after inserting the data. A lower setting affects all tables on the server and results in unused pages in all tables.

Fast bulk copy ignores the **page utilization percent** setting and always allocates new extents until there are no more extents available in the database.

Error Log

The parameters in this group configure the Adaptive Server error log and the logging of Adaptive Server events to the Windows NT event log.

event log computer name (Windows NT only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	'LocalSystem'
Valid values	• Name of an NT machine on the network configured to record Adaptive Server messages • 'LocalSystem' • 'NULL'
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **event log computer name** parameter specifies the name of the Windows NT PC that logs Adaptive Server messages in its Windows NT Event Log. You can use this parameter to have Adaptive Server messages logged to a remote machine. This feature is available on Windows NT servers only.

A value of 'LocalSystem' or 'NULL' specifies the default local system.

You can also use the Server Config utility to set the **event log computer name** parameter by specifying the Event Log Computer Name under Event Logging.

Setting the **event log computer name** parameter with **sp_configure** or specifying the Event Log Computer Name under Event Logging overwrites the effects of the command-line **-G** option, if it was specified. If Adaptive Server was started with the **-G** option, you can change the destination remote machine by setting the **event log computer name** parameter.

For more information about logging Adaptive Server messages to a remote site, see *Configuring Adaptive Server for Windows NT*.

event logging (Windows NT only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **event logging** parameter enables and disables the logging of Adaptive Server messages in the Windows NT Event Log. This feature is available on Windows NT servers only.

A value of 1 enables Adaptive Server message logging in the Windows NT Event Log; a value of 0 disables it. The default is event logging enabled.

You use the Server Config utility to set the **event logging** parameter by selecting "Use Windows NT Event Logging" under Event Logging.

Setting the **event logging** parameter or selecting "Use Windows NT Event Logging" overwrites the effects of the command-line **-g** option, if it was specified.

log audit logon failure

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **log audit logon failure** parameter specifies whether to log unsuccessful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.

A value of 1 requests logging of unsuccessful logins; a value of 0 specifies no logging.

log audit logon success

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **log audit logon success** parameter specifies whether to log successful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.

A value of 1 requests logging of successful logins; a value of 0 specifies no logging.

Extended Stored Procedures

The parameters in this group affect the behavior of extended stored procedures (ESPs).

esp execution priority

Summary Information	
Name in pre-11.0 release	N / A
Default value	8
Range of values	0–15
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **esp execution priority** parameter sets the priority of the XP Server thread for ESP execution. ESPs can be CPU-intensive over long periods of time. Also, since XP Server resides on the same machine as Adaptive Server, XP Server can impact Adaptive Server's performance.

Use **esp execution priority** to set the priority of the XP Server thread for ESP execution. See the *Open Server Server-Library/C Reference Manual* for information about scheduling Open Server threads.

esp execution stacksize

Summary Information	
Name in pre-11.0 release	N / A
Default value	34816
Range of values	34816–2^{32}
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **esp execution stacksize** parameter sets the size of the stack, in bytes, to be allocated for ESP execution.

Use this parameter if you have your own ESP functions that require a larger stack size than the default, 34816.

esp unload dll

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **esp unload dll** parameter specifies whether DLLs that support ESPs should be automatically unloaded from XP Server memory after the ESP call has completed.

If **esp unload dll** is set to 0, DLLs are not automatically unloaded. If it is set to 1, they are automatically unloaded.

If **esp unload dll** is set to 0, you can still unload individual DLLs explicitly at run time, using the **sp_freedll** system procedure.

start mail session (Windows NT only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **start mail session** parameter enables and disables the automatic initiation of an Adaptive Server mail session when Adaptive Server is started. This feature is available on Windows NT servers only.

A value of 1 configures Adaptive Server to start a mail session the next time Adaptive Server is started. A value of 0 configures Adaptive Server not to start a mail session at the next restart.

If **start mail session** is 0, you can start an Adaptive Server mail session explicitly, using the **xp_startmail** system ESP.

Before setting the **start mail session** parameter, you must prepare your Windows NT system by creating a mailbox and mail profile for Adaptive Server. Then, you must create an Adaptive Server account for Sybmail. See *Configuring Adaptive Server for Windows NT* for information about preparing your system for Sybmail.

xp_cmdshell context

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Valid values	0, 1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **xp_cmdshell context** parameter sets the security context for the operating system command to be executed using the **xp_cmdshell** system ESP.

Setting **xp_cmdshell context** to 1 restricts the **xp_cmdshell** security context to users who have accounts at the operating system level. Its behavior is platform-specific. If **xp_cmdshell context** is set to 1, then in order to use an **xp_cmdshell** ESP, an operating system user account must exist for the Adaptive Server user name. For example, an Adaptive Server user named "sa" will not be able to use **xp_cmdshell** unless he or she has an operating system level user account named "sa".

On Windows NT, when **xp_cmdshell context** is set to 1, **xp_cmdshell** succeeds only if the user name of the user logging into Adaptive Server is a valid Windows NT user name with Windows NT system administration privileges on the system on which Adaptive Server is running.

On other platforms, when **xp_cmdshell context** is set to 1, **xp_cmdshell** succeeds only if Adaptive Server was started by a user with "superuser" privileges at the operating system level. When Adaptive Server gets a request to execute **xp_cmdshell**, it checks the *uid* of the user name of the ESP requestor and runs the operating system command with the permissions of that *uid*.

If **xp_cmdshell context** is 0, the permissions of the operating system account under which Adaptive Server is running are the permissions used to execute an operating system command from **xp_cmdshell**. This allows users to execute operating commands that they would not ordinarily be able to execute under the security context of their own operating system accounts..

General Information

The parameters in this group are not related to any particular area of Adaptive Server behavior.

configuration file

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	N / A
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **configuration file** parameter specifies the location of the configuration file currently in use. See “Using sp_configure with a Configuration File” on page 11-10 for a complete description of configuration files.

Note that in **sp_configure** output the “Run Value” column displays only 10 characters. For this reason, the output may not display the entire path and name of your configuration file.

Languages

The parameters in this group configure languages, sort orders, and character sets.

default character set id

Summary Information	
Name in pre-11.0 release	**default character set id**
Default value	1
Range of values	0–255
Status	Static
Display level	Intermediate
Required role	System Administrator

The **default character set id** parameter specifies the number of the default character set used by the server. The default is set at installation time, and can be changed later with the Sybase installation utilities. See Chapter 13, "Configuring Character Sets, Sort Orders, and Languages," for a discussion of how to change character sets and sort orders.

default language id

Summary Information	
Name in pre-11.0 release	**default language**
Default value	0
Range of values	0–32767
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **default language id** parameter is the number of the language that is used to display system messages unless a user has chosen another language from those available on the server. us_english always has an ID of 0. Additional languages are assigned unique numbers as they are added.

default sortorder id

Summary Information	
Name in pre-11.0 release	**default sortorder id**
Default value	50
Range of values	0–255
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **default sortorder id** parameter is the number of the sort order that is installed as the default on the server. If you want to change the default sort order, see Chapter 13, "Configuring Character Sets, Sort Orders, and Languages."

disable character set conversions

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (enabled)
Valid values	0 (enabled), 1 (disabled)
Status	Static
Display level	Comprehensive
Required role	System Administrator

Changing **disable character set conversions** to 1 turns off character set conversion for data moving between clients and Adaptive Server. By default, Adaptive Server performs conversion on data moving to and from clients that use character sets that are different than the server's. For example, if some clients use Latin-1 (iso_1) and Adaptive Server uses Roman-8 (roman8) as its default character set, data from the clients is converted to Roman-8 when being loaded into Adaptive Server. For clients using Latin-1, the data is reconverted when it is sent to the client; for clients using the same character set as Adaptive Server, the data is not converted.

By setting **disable character set conversions**, you can request that no translation take place. For example, if all clients are using a given character set, and you want Adaptive Server to store all data in that character set, you can set **disable character set conversions** to 1, and no conversion will take place.

number of languages in cache

Summary Information	
Name in pre-11.0 release	**language in cache**
Default value	3
Range of values	3–100
Status	Static
Display level	Intermediate
Required role	System Administrator

The **number of languages in cache** parameter indicates the maximum number of languages that can be held simultaneously in the language cache. The default is 3.

Lock Manager

The parameters in this group configure locks.

address lock spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the **address lock spinlock ratio** sets the number of rows in the internal address locks hash table that are protected by one spinlock.

Adaptive Server manages the acquiring and releasing of address locks using an internal hash table with 1031 rows (known as hash buckets). This table can use one or more spinlocks to serialize access between processes running on different engines.

Adaptive Server's default value for **address lock spinlock ratio** is 100, which defines 11 spinlocks for the address locks hash table. The first 10 spinlocks protect 100 rows each, and the eleventh spinlock protects the remaining 31 rows. If you specify a value of 1031 or greater for **address lock spinlock ratio**, Adaptive Server uses only 1 spinlock for the entire table.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

deadlock checking period

Summary Information	
Name in pre-11.0 release	N / A
Default value	500
Range of values	0–2147483
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **deadlock checking period** parameter specifies the minimum amount of time (in milliseconds) before Adaptive Server initiates a deadlock check for a process that is waiting on a lock to be released. Deadlock checking is time-consuming overhead for applications that experience no deadlocks or very few deadlocks, and the overhead grows as the percentage of lock requests that must wait for a lock also increases.

If you set this value to a nonzero value (*n),* Adaptive Server initiates a deadlock check after a process waits at least *n* ms. For example, you can make a process wait at least 700 ms. for a lock before each deadlock check as follows:

```
sp_configure "deadlock checking period", 700
```

If you set this parameter to 0, Adaptive Server initiates deadlock checking at the time each process begins to wait for a lock. Any value less than the number of milliseconds in a clock tick is treated as 0. See “sql server clock tick length” on page 11-126 for more information.

Configuring **deadlock checking period** to a higher value produces longer delays before deadlocks are detected. However, since Adaptive Server grants most lock requests before this time elapses, the deadlock checking overhead is avoided for those lock requests. If your applications deadlock infrequently, set **deadlock checking period** to a higher value to avoid the overhead of deadlock checking for most processes. Otherwise, the default value of 500 ms. should suffice.

Use **sp_sysmon** to determine the frequency of deadlocks in your system and the best setting for the **deadlock checking period** parameter. See “Lock Management” on page 24-59 in the *Performance and Tuning Guide* for more information.

deadlock retries

Summary Information	
Name in pre-11.0 release	N / A
Default value	5
Range of values	0–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **deadlock retries** parameter specifies the number of times a transaction can attempt to acquire a lock when deadlocking occurs during an index page split or shrink.

For example, see Figure 11-2, which illustrates the following scenario:

- Transaction A locks page 1007 and needs to acquire a lock on page 1009 in order to update the page pointers for a page split.
- Transaction B is also inserting an index row that causes a page split, holds a lock on page 1009, and needs to acquire a lock on page 1007.

In this situation, rather than immediately choosing a process as a deadlock victim, Adaptive Server relinquishes the index locks for one of the transactions. This often allows the other transaction to complete and release its locks.

For the transaction that surrendered its locking attempt, the index is rescanned from the root page, and the page split operation is attempted again, up to the number of times specified by **deadlock retries**.

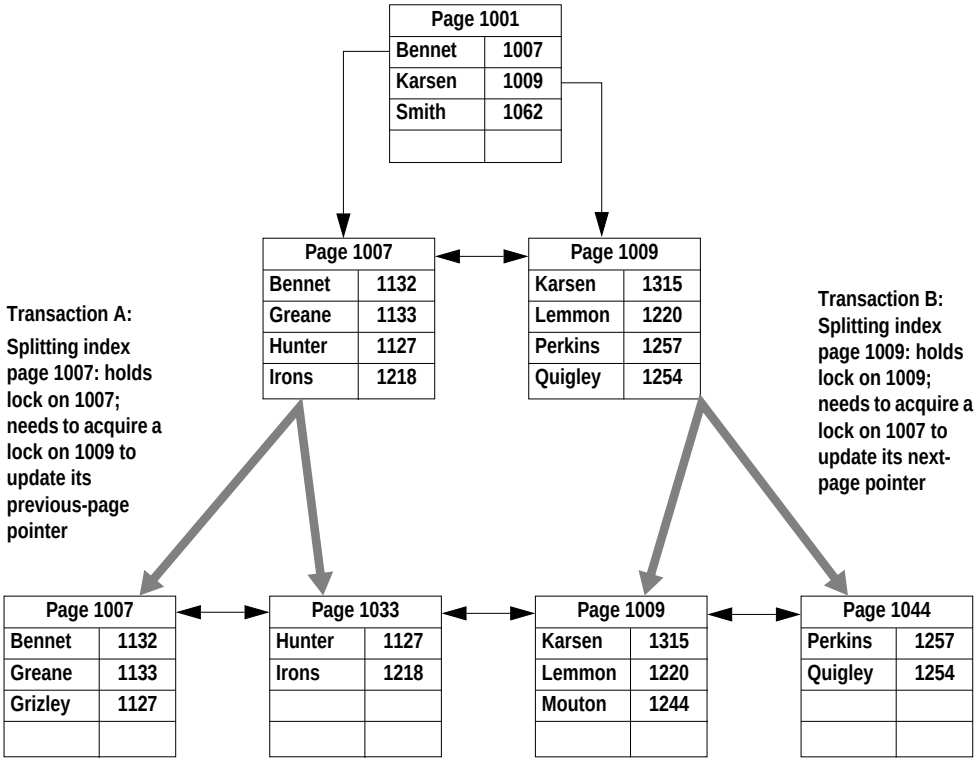


Figure 11-2: Deadlocks during page splitting in a clustered index

The system procedure **sp_sysmon** reports on deadlocks and deadlock retries. See "Retries and Deadlocks" on page 24-55 of the *Performance and Tuning Guide*.

freelock transfer block size

Summary Information	
Name in pre-11.0 release	N / A
Default value	30
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

When a process running on a multi-engine Adaptive Server requests a lock, Adaptive Server looks for one in its engine's freelock list. If the engine freelock list is out of locks, Adaptive Server moves a certain number of locks from its global freelock list to the engine freelock list. After a process completes, the locks released by those processes accumulate in the engine's freelock list. When the number of locks in the engine list reaches its maximum (defined by the **max engine freelocks** parameter), Adaptive Server moves a number of locks from the engine freelock list to the global freelock list. This replenishes the number of locks available to other engines from the global list.

The **freelock transfer block size** configuration parameter specifies the number of locks moved between the engine freelock lists and the global freelock list. Adaptive Server's default value for **freelock transfer block size** is 30, and the minimum value allowed is 1. For example, you change the transfer size to 50 locks as follows:

```
sp_configure "freelock transfer block size", 50
```

When you set this to a higher value, the frequency of transfers between the engine freelock lists and the global freelock list is reduced, which lowers the contention in accessing the global freelock list. However, a higher value can result in accessing many more lock structures than a process needs. The maximum value allowed for **freelock transfer block size** cannot exceed more than half the maximum number of locks available to an engine's freelock list, as defined by the following formula:

((**max engine freelocks** percent value * **number of locks** value) / **max online engines** value) / 2

For example, if **max engine freelocks** is set to 10 percent and the **number of locks** value is 5000, the maximum value allowed for **freelock transfer block size** is 50 for an Adaptive Server running with 5 engines.

If you try to set **freelock transfer block size** to a value that is higher than its maximum value, Adaptive Server returns an error message and leaves the parameter unchanged. It also returns an error if the current **freelock transfer block size** value will exceed the maximum when you attempt to increase **max online engines** or decrease either **max engine freelocks** or **number of locks**. Adaptive Server sets this maximum to avoid draining the engine freelock lists of too many locks, which can force it to get more locks immediately from the global freelocks list.

max engine freelocks

Summary Information	
Name in pre-11.0 release	N / A
Default value	10
Range of values	1–50
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

When a process running on a multi-engine Adaptive Server requests a lock, it looks for one in its engine's freelock list. If the engine freelock list is out of locks, Adaptive Server moves a certain number of locks (defined by the **freelock transfer block size** parameter) from its global freelock list to the engine freelock list. The total number of locks in the global freelock list is defined by the **number of locks** parameter value. For a single engine Adaptive Server, the entire global freelock list is moved to the engine freelock list at server start-up, regardless of the value of this parameter.

After an engine completes a process, all the locks held by that process are released and returned to that engine's freelock list. This reduces the contention of each engine accessing the global freelock list. However, if the number of locks released to the engine exceed the maximum number of locks allowed in the engine's freelock list, Adaptive Server moves a number of locks (defined by **freelock transfer block size**) to the global freelock list. This replenishes the number of locks available to other engines from the global list.

You can specify the maximum number of locks available to the engine freelock lists as a percentage of the total number of locks available to your server by using the **max engine freelocks** configuration parameter. You can specify any value from 1 percent to 50 percent. For example, 20 percent of the total number of locks can be the maximum number of locks available to the engine freelock lists, as follows:

```
sp_configure "max engine freelocks", 20
```

If your server has 5000 locks configured for the **number of locks** parameter, 20 percent (or 1000) of those locks represents the maximum number of locks available to each engine freelock list. The maximum for each engine freelock list depends on the number of engines configured for Adaptive Server (**max online engines** parameter). If your server has 5 engines, the maximum for each engine freelock list is 1000 divided by 5, or 200 locks. Therefore, a change in the **number of locks** or **max online engines** configuration parameter can affect the maximum for each engine freelock list, even if the value for **max engine freelocks** remains the same.

For some servers, if you set **max engine freelocks** too high, most of the available locks may end up in each engine's freelock list, leaving very few locks in Adaptive Server's global freelock list. If an engine's freelock list becomes empty, it is likely that the global freelock list is also empty, which results in Adaptive Server error message 1279, even though other engines have locks in their freelock lists. Error message 1279 reads as follows:

```
SQL Server has run out of locks on engine %d. Re-
run your command when there are fewer active
users, or contact a user with System Administrator
(SA) role to reconfigure max engine freelocks.
```

You should either decrease the configured value for **max engine freelocks** or increase the configured value of **number of locks** to avoid frequent occurrences of message 1279. This message differs from message 1204, which indicates that Adaptive Server has no more locks in either the global freelock list or the engine freelock lists.

number of locks

Summary Information	
Name in pre-11.0 release	**locks**
Default value	5000
Range of values	1000–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of locks** parameter sets the total number of available locks for all users on Adaptive Server.

The total number of locks needed by Adaptive Server depends on the number and nature of the queries that are running. The number of locks required by a query can vary widely, depending on the number of concurrent and parallel processes and the types of actions performed by the transactions. To see how many locks are in use at a particular time, use the **sp_lock** stored procedure.

Configuring the correct number of locks is a matter of experience and familiarity with the system. For serial operation, you can start with an arbitrary number of 20 locks for each active, concurrent connection.

Parallel execution requires more locks than serial execution. For example, if you find that queries use an average of five worker processes, try increasing, by one-third, the value of **number of locks** configured for serial operation.

If **number of locks** is set too low, and the system runs out of locks, Adaptive Server displays a server-level error message. If users report lock errors, it typically indicates that you need to increase **number of locks**; but remember that locks use memory. See “Number of Locks” on page 8-13 for information.

page lock spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the **page lock spinlock ratio** configuration parameter sets the number of rows in the internal page locks hash table that are protected by one **spinlock**.

Adaptive Server manages the acquiring and releasing of page locks using an internal hash table with 1031 rows (known as hash buckets). This table can use one or more spinlocks to serialize access between processes running on different engines.

Adaptive Server's default value for **page lock spinlock ratio** is 100, which defines 11 spinlocks for the page locks hash table. The first 10 spinlocks protect 100 rows each, and the eleventh spinlock protects the remaining 31 rows. If you specify a value of 1031 or greater for **page lock spinlock ratio**, Adaptive Server uses only 1 spinlock for the entire table.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

table lock spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	20
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the **table lock spinlock ratio** configuration parameter sets the number of rows in the internal table locks hash table that are protected by one **spinlock**.

Adaptive Server manages the acquiring and releasing of table locks using an internal hash table with 101 rows (known as hash buckets). This table can use one or more spinlocks to serialize access between processes running on different engines.

Adaptive Server's default value for **table lock spinlock ratio** is 20, which defines 6 spinlocks for the table locks hash table. The first 5 spinlocks protect 20 rows each; the sixth spinlock protects the last row. If you specify a value of 101 or greater for **table lock spinlock ratio**, Adaptive Server uses only 1 spinlock for the entire table.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

Memory Use

The following parameter optimizes Adaptive Server's memory use:

executable codesize + overhead

Summary Information	
Name in pre-11.0 release	**sql server code size**
Default value	N / A
Range of values	N / A
Status	Calculated
Display level	Basic
Required role	System Administrator

The **executable codesize + overhead** parameter reports the combined size (in kilobytes) of the Adaptive Server executable and overhead. It is a calculated value and is not user-configurable. For more information, see "System Procedures for Configuring Memory" on page 8-4.

Metadata Caches

The following parameters help set the metadata cache size for frequently used system catalog information. The **metadata cache** is a reserved area of memory used for tracking information on databases, indexes, or objects. The greater the number of open databases, indexes, or objects, the larger the metadata cache size. For a discussion of metadata caches in a memory-usage context, see "Open Databases, Open Indexes, and Open Objects" on page 8-12.

number of open databases

Summary Information	
Name in pre-11.0 release	**open databases**
Default value	12
Range of values	5–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of open databases** parameter sets the maximum number of databases that can be open simultaneously on Adaptive Server.

When you calculate a value for the **number of open databases** parameter, include the system databases *master*, *model*, *sybsystemprocs*, and *tempdb*. Also, if you have installed auditing, include the *sybsecurity* database. Also, count the sample databases *pubs2* and *pubs3*, the syntax database *sybsyntax*, and the **dbcc** database *dbccdb* if they are installed.

If you are planning to make a substantial change, such as loading a large database from another server, you can calculate an estimated metadata cache size by using the system procedure **sp_helpconfig**. **sp_helpconfig** displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. A database metadata descriptor represents the state of the database while it is in use or cached between uses.

Optimizing the *number of open databases* Parameter for Your System

The default run value for **number of open databases** is 12. If Adaptive Server displays a message saying that you have exceeded the allowable number of open databases, you will need to adjust this value.

In order to set the **number of open databases** parameter optimally, perform the following steps:

- Step 1: Determine the total number of databases (database metadata descriptors).
- Step 2: Reset the **number of open databases** parameter to that number.

- Step 3: Restart Adaptive Server.
- Step 4: Find the number of active databases (active metadata descriptors) during a peak period.
- Step 5: Reset the **number of open databases** parameter to that number, plus 10 percent.
- Step 6: Restart Adaptive Server.

The following section details the basic steps listed above.

1. Use the **sp_countmetadata** system procedure to find the total number of database metadata descriptors. For example:

   ```
   sp_countmetadata "open databases"
   ```

   The best time to run **sp_countmetadata** is when there is little activity on the server. Running **sp_countmetadata** during a peak time can cause contention with other processes, depending on the number of other databases in the server, since **sp_countmetadata** has to access the system tables.

   Suppose Adaptive Server reports the following information:

   ```
   There are 50 databases, requiring 1719 Kbytes of
   memory. The 'open databases' configuration
   parameter is currently set to 500.
   ```

2. Configure the **number of open databases** parameter with this new value (50):

   ```
   sp_configure "number of open databases", 50
   ```

3. Restart the server.

   This new configuration is only a start; the ideal size should be based on the number of **active** metadata database cache descriptors, not the **total** number of databases.

4. During a peak period, find the number of active metadata descriptors, for example:

   ```
   sp_monitorconfig "open databases"
   ```

```
Usage information at date and time: Jan 14 1997  8:54AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------  ---------------   -------
number of open  50        20       40.00     26                No
databases
```

At this peak period, 20 metadata database descriptors are active; the maximum number of descriptors that have been active since the server was last started is 26.

See **sp_monitorconfig** in the *Adaptive Server Reference Manual* for more information about this system procedure.

5. Configure the **number of open databases** parameter to this value (26), plus additional space for 10 percent more (about 3, for a total of 29):

   ```
   sp_configure "number of open databases", 29
   ```

6. Restart the server.

If there is a lot of activity on the server, for example, if databases are being added or dropped, run **sp_monitorconfig** periodically. You will need to reset the cache size as the number of active descriptors changes. However, avoid changing the **number of open databases** configuration parameter too often, since you will need to restart Adaptive Server each time, before the parameter can take effect.

number of open indexes

Summary Information	
Name in pre-11.0 release	N / A
Default value	500
Range of values	100–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of open indexes** parameter sets the maximum number of indexes that can be used simultaneously on Adaptive Server.

If you are planning to make a substantial change, such as loading databases with a large number of indexes from another server, you can calculate an estimated metadata cache size by using the system procedure **sp_helpconfig. sp_helpconfig** displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. An index metadata descriptor represents the state of an index while it is in use or cached between uses.

Optimizing the *number of open indexes* Parameter for Your System

The default run value is 500. If this number is insufficient, Adaptive Server displays a message after trying to reuse active index descriptors, and you will need to adjust this value.

In order to configure the **number of open indexes** parameter optimally, perform the following steps:

- Step 1: Determine the total number of indexes (total number of index metadata descriptors).
- Step 2: Reset the **number of open indexes** parameter to that number.
- Step 3: Restart Adaptive Server.
- Step 4: Find the number of active indexes (index metadata. descriptors) during a peak period.
- Step 5: Reset the **number of open indexes** parameter to that number, plus 10 percent.
- Step 6: Restart Adaptive Server.

The following section details the basic steps listed above:

1. Use the **sp_countmetadata** system procedure to find the total number of index metadata descriptors. For example:

   ```
   sp_countmetadata "open indexes"
   ```

   The best time to run **sp_countmetadata** is when there is little activity in the server. Running **sp_countmetadata** during a peak time can cause contention with other processes depending on the number of other databases in the server, since **sp_countmetadata** has to access the system tables.

   Suppose Adaptive Server reports the following information:

   ```
   There are 698 user indexes in all database(s),
   requiring 286.289 Kbytes of memory. The 'open
   indexes' configuration parameter is currently set
   to 500.
   ```

2. Configure the **number of open indexes** parameter to this new value (698) as follows:

   ```
   sp_configure "number of open indexes", 698
   ```

3. Restart the server.

   This new configuration is only a start; the ideal size should be based on the number of *active* index metadata cache descriptors, not the total number of indexes.

4. During a peak period, find the number of active index metadata descriptors, for example:

   sp_monitorconfig "open indexes"

```
Usage information at date and time: Jan 14 1997 8:54AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  182       516      73.92      590               No
indexes
```

   In this example, 590 is the maximum number of index descriptors that have been used since the server was last started.

   See **sp_monitorconfig** in the *Adaptive Server Reference Manual* for more information about this system procedure.

5. Configure the **number of open indexes** configuration parameter to this value (590), plus additional space for 10 percent more (59), for a total of 649:

   sp_configure "number of open indexes", 649

6. Restart the server.

If there is a lot of activity on the server, for example, if tables are being added or dropped, run **sp_monitorconfig** periodically. You will need to reset the cache size as the number of active descriptors changes. However, avoid changing the **number of open indexes** configuration parameter too often, since you need to restart Adaptive Server each time before the parameter can take effect.

number of open objects

Summary Information	
Name in pre-11.0 release	**open objects**
Default value	500
Range of values	100–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of open objects** parameter sets the maximum number of objects that can be open simultaneously on Adaptive Server.

If you are planning to make a substantial change, such as loading databases with a large number of objects from another server, you can calculate an estimated metadata cache size by using the system procedure **sp_helpconfig. sp_helpconfig** displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. An object metadata descriptor represents the state of an object while it is in use, or cached between uses.

Optimizing the *number of open objects* Parameter for Your System

The default run value is 500. If this number is insufficient, Adaptive Server displays a message after trying to re-use active object descriptors. You will need to adjust this value.

To set the **number of open objects** parameter optimally, perform the following steps:

- Step 1: Determine the total number of objects (total number of object metadata descriptors).
- Step 2: Reset the **number of open objects** parameter to that number.
- Step 3: Restart Adaptive Server.
- Step 4: Find the number of active objects (active metadata descriptors) during a peak period.
- Step 5: Reset the **number of open objects** parameter to that number plus 10 percent.
- Step 6: Restart Adaptive Server.

The following section details the basic steps listed above:

1. Use the **sp_countmetadata** system procedure to find the total number of object metadata cache descriptors. For example:

   ```
   sp_countmetadata "open objects"
   ```

   The best time to run **sp_countmetadata** is when there is little activity in the server. Running **sp_countmetadata** during a peak time can cause contention with other processes, since **sp_countmetadata** has to access the system tables.

   Suppose Adaptive Server reports the following information:

   ```
   There are 340 user objects in all database(s),
   requiring 140.781 Kbytes of memory. The 'open
   objects' configuration parameter is currently set
   to 500
   ```

2. Configure the number of open objects parameter to that value, as follows:

   sp_configure "number of open objects", 357

   357 covers the 340 user objects, plus 5 percent to accommodate temporary tables.

3. Restart the server.

   This new configuration is only a start; the ideal size should be based on the number of *active* object metadata cache descriptors, not the *total* number of objects.

4. During a peak period, find the number of active metadata cache descriptors, for example:

   sp_monitorconfig "open objects"

```
Usage information at date and time: Jan 14 1997 8:54AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  160       357      71.40      397               No
objects
```

   In this example, 397 is the maximum number of object descriptors that have been used since the server was last started.

5. Configure the **number of open objects** to this number (397), plus 10 percent (40), for a total of 437:

   sp_configure "number of open objects", 437

6. Restart the server.

If there is a lot of activity on the server, for example, if tables are being added or dropped, run **sp_monitorconfig** periodically. You will need to reset the cache size as the number of active descriptors changes. However, avoid changing the **number of open objects** configuration parameter too often, since you need to restart Adaptive Server before it can take effect. See **sp_monitorconfig** in the *Adaptive Server Reference Manual* for more information.

open index hash spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **open index hash spinlock ratio** parameter sets the number of index metadata descriptor hash tables that are protected by one **spinlock**. This configuration parameter is used for multiprocessing systems only.

All the index descriptors belonging to the table are accessible through a hash table. When a query is run on the table, Adaptive Server uses hash tables to look up the necessary index information in its *sysindexes* rows. A hash table is an internal mechanism used by Adaptive Server to retrieve information quickly.

Usually, you do not need to change this parameter. In rare instances, however, you may need to reset it if Adaptive Server demonstrates contention from hash spinlocks. You can get information about spinlock contention by using the system procedure **sp_sysmon**. For more about **sp_sysmon**, see Chapter 24, "Monitoring Performance with sp_sysmon" in the *Performance and Tuning Guide*.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

open index spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–214748364
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **open index spinlock ratio** parameter specifies the number of index metadata descriptors that are protected by one **spinlock**.

Adaptive Server uses a spinlock to protect an index descriptor, since more than one process can access the contents of the index descriptor. This configuration parameter is used for multiprocessing systems only.

The value specified for this parameter defines the ratio of index descriptors per spinlock. The default value for this parameter is 100, one spinlock for each 100 index descriptors configured for your server.

If one spinlock is shared by too many index descriptors, it can cause spinlock contention. Use **sp_sysmon** to get a report on spinlock contention. See Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide* more information about spinlock contention. If **sp_sysmon** output indicates an index descriptor spinlock contention of more than 3 percent, try decreasing the value of the **open index spinlock ratio** parameter.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

open object spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **open object spinlock ratio** parameter specifies the number of object descriptors that are protected by one **spinlock**. Adaptive Server uses a spinlock to protect an object descriptor, since more than one process can access the contents of the object descriptor. This configuration parameter is used for multiprocessing systems only.

The default value for this parameter is 100; 1 spinlock for each 100 object descriptors configured for your server. If your server is configured with only one engine, Adaptive Server sets only 1 object descriptor spinlock, regardless of the number of object descriptors.

If one spinlock is shared by too many object descriptors, it causes spinlock contention. Use **sp_sysmon** to get a report on spinlock contention. See Chapter 24, "Monitoring Performance with sp_sysmon" in the *Performance and Tuning Guide* for more information on spinlock contention. If **sp_sysmon** output indicates an object descriptor spinlock contention of more than 3 percent, try decreasing the value of the **open object spinlock ratio** parameter.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

Network Communication

Parameters in this group configure communication between Adaptive Server and remote servers, and between Adaptive Server and client programs.

allow remote access

Summary Information	
Name in pre-11.0 release	**remote access**
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **allow remote access** parameter controls logins from remote Adaptive Servers. The default value is 1 so that Adaptive Server can communicate with Backup Server. Only a System Security Officer can set **allow remote access**.

Setting the value to 0 disables server-to-server remote procedure calls. Since Adaptive Server communicates with Backup Server via remote procedure calls, setting this parameter to 0 makes it impossible to back up a database.

Since other system administration actions are required to enable remote servers other than Backup Server to execute remote procedure calls, leaving this option set to 1 does not constitute a security risk.

default network packet size

Summary Information	
Name in pre-11.0 release	**default network packet size**
Default value	512
Range of values	512–524288
Status	Static
Display level	Intermediate
Required role	System Administrator

The **default network packet size** parameter configures the default packet size for all Adaptive Server users. The default value is 512 bytes. You

can set **default network packet size** to any multiple of 512 bytes up to a maximum of 524,288 bytes. Values that are not even multiples of 512 are rounded down.

Memory for all users who log in with the default packet size is allocated from Adaptive Server's memory pool, as set with the **total memory** configuration parameter. This memory is allocated for network packets when Adaptive Server is started.

Each Adaptive Server user connection uses:

- One read buffer
- One buffer for messages
- One write buffer

Each of these buffers requires **default network packet size** bytes. The total amount of memory allocated for network packets is:

(**number of user connections** + **number of worker processes**) * 3 * **default network packet size**

For example, if you set the **default network packet size** to 1024 bytes, and you have 50 user connections and 20 worker processes, the amount of network memory required is:

(50 + 20) * 3 * 1024 = 215040 bytes

If you increase the **default network packet size**, you must also increase the **max network packet size** to at least the same size. If the value of **max network packet size** is greater than the value of **default network packet size**, you need to increase the value of **additional network memory**. See "additional network memory" on page 11-89 for further information.

Use **sp_sysmon** to see how changing the **default network packet size** parameter affects network I/O management and task switching. For example, try increasing **default network packet size** and then checking **sp_sysmon** output to see how this affects **bcp** for large batches. See "Network Packet Received" on page 24-28, "Network Packet Sent" on page 24-29, and "Network I/O Management" on page 24-94 in the *Performance and Tuning Guide*.

max network packet size

Summary Information	
Name in pre-11.0 release	**maximum network packet size**
Default value	512
Range of values	512–524288
Status	Static
Display level	Intermediate
Required role	System Administrator

The **max network packet size** parameter specifies the maximum network packet size that can be requested by clients communicating with Adaptive Server.

If some of your applications send or receive large amounts of data across the network, these applications can achieve significant performance improvement by using larger packet sizes. Two examples are large bulk copy operations and applications that read or write large *text* or *image* values.

Generally, you want:

- The value of **default network packet size** to be small for users who perform short queries
- The **max network packet size** configuration parameter to be large enough to allow users who send or receive large volumes of data to request larger packet sizes

The default value for **max network packet size** is 512 bytes. It must always be as large as, or larger than, the **default network packet size**. Values that are not even multiples of 512 are rounded down.

In order for client applications that explicitly request a larger network packet size to receive it, you must also configure additional network memory. See “additional network memory” on page 11-89 for more information about the relationship between **max network packet size** and **additional network memory.**

See **bcp** and **isql** in the *Utility Programs* manual for your platform manual for information on using larger packet sizes from these programs. Open Client Client-Library documentation includes information on using variable packet sizes.

Choosing Packet Sizes

The **default network packet size** and **max network packet size** parameters must be multiples of 512. In addition, for best performance, choose a server packet size that works efficiently with the underlying packet size on your network. The two goals are:

- Reducing the number of server reads and writes to the network
- Reducing unused space in network packets (increasing network throughput)

For example, if your network packet size carries 1500 bytes of data, setting Adaptive Server's packet size to 1024 (512*2) will probably achieve better performance than setting it to 1536 (512*3).
Figure 11-3 shows how four different packet size configurations would perform in such a scenario.

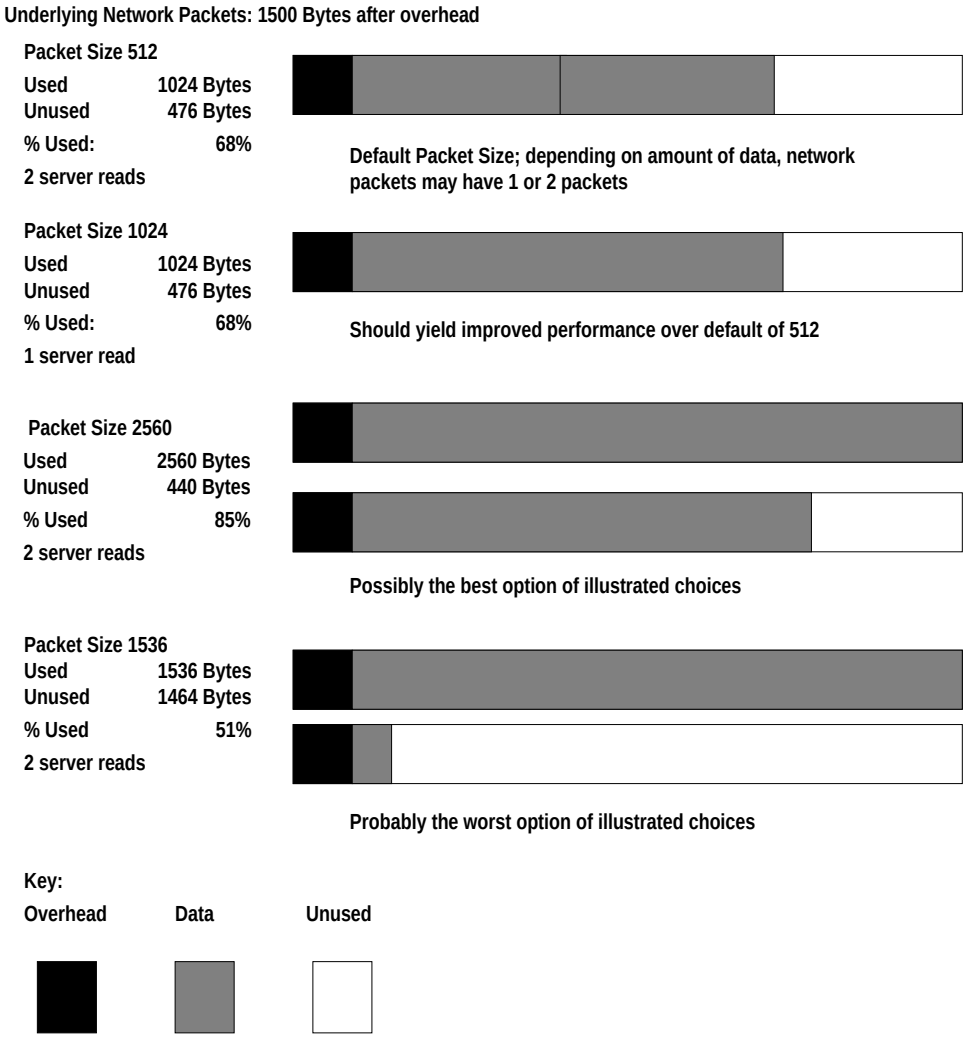


Figure 11-3: Factors in determining packet size

After you determine the available data space of the underlying packets on your network, you should perform your own benchmark tests to determine the optimum size for your configuration.

Use **sp_sysmon** to see how changing the **max network packet size** parameter affects network I/O management and task switching. For example, try increasing **max network packet size** and then checking

sp_sysmon output to see how this affects **bcp** for large batches. See "Network Packet Received" on page 24-28, "Network Packet Sent" on page 24-29, and "Network I/O Management" on page 24-94 in the *Performance and Tuning Guide*.

max number network listeners

Summary Information	
Name in pre-11.0 release	**cmaxnetworks**
Default value	5
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **max number network listeners** parameter specifies the maximum number of network listeners allowed by Adaptive Server at one time.

Each master port has one network listener. Generally, there is no need to have multiple master ports, unless your Adaptive Server needs to communicate over more than one network type. Some platforms support both socket and TLI (Transport Layer Interface) network interfaces. Refer to the configuration documentation for your platform for information on supported network types. If your Adaptive Server does need to use multiple network types, configure **max number network listeners** to accommodate the number of network types you need.

number of remote connections

Summary Information	
Name in pre-11.0 release	**remote connections**
Default value	20
Range of values	0–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

The **number of remote connections** parameter specifies the number of logical connections that can be open to and from an Adaptive Server at one time. Each simultaneous connection to XP Server for ESP execution uses up to one remote connection each. For more information, see Chapter 7, "Managing Remote Servers," in the *Security Administration Guide.*

number of remote logins

Summary Information	
Name in pre-11.0 release	**remote logins**
Default value	20
Range of values	0–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

The **number of remote logins** parameter controls the number of active user connections from Adaptive Server to remote servers. Each simultaneous connection to XP Server for ESP execution uses up to one remote login each. You should set this parameter to the same (or a lower) value as **number of remote connections**. For more information, see Chapter 7, "Managing Remote Servers," in the *Security Administration Guide.*

number of remote sites

Summary Information	
Name in pre-11.0 release	**remote sites**
Default value	10
Range of values	0–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

The **number of remote sites** parameter determines the maximum number of remote sites that can access Adaptive Server simultaneously. An Adaptive Server-to-XP Server connection uses one remote site connection.

Internally, **number of remote sites** determines the number of site handlers that can be active at any one time; all server accesses from a single site are managed with a single site handler. For more information, see Chapter 7, "Managing Remote Servers," in the *Security Administration Guide.*

remote server pre-read packets

Summary Information	
Name in pre-11.0 release	**pre-read packets**
Default value	3
Range of values	0–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

The **remote server pre-read packets** parameter determines the number of packets that will be "pre-read" by a site handler during connections with remote servers.

All communication between two servers is managed through a single site handler, to reduce the required number of connections.

The site handler can pre-read and keep track of data packets for each user process before the receiving process is ready to accept them.

The default value for **remote server pre-read packets** is 3, which is appropriate for most servers. Increasing the value uses more memory; decreasing the value can slow network traffic between servers. For more information, see Chapter 7, "Managing Remote Servers," in the *Security Administration Guide*.

tcp no delay

Summary Information	
Name in pre-11.0 release	**T1610** (trace flag)
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **tcp no delay** parameter controls TCP (Transmission Control Protocol) packet batching. The default value is 0, which means that TCP packets are batched.

TCP normally batches small logical packets into single larger physical packets (by briefly delaying packets) in order to fill physical network frames with as much data as possible. This is intended to improve network throughput in terminal emulation environments where there are mostly keystrokes being sent across the network.

However, applications that use small TDS (Tabular Data Stream™) packets may benefit from disabling TCP packet batching. To disable TCP packet batching, set **tcp no delay** to 1.

➤ ***Note***

Disabling TCP packet batching means that packets will be sent, regardless of size; this will increase the volume of network traffic.

O/S Resources

The parameters in this group are related to Adaptive Server's use of operating system resources.

max async i/os per engine

Summary Information	
Name in pre-11.0 release	**cnmaxaio_engine**
Default value	2147483647
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **max async i/os per engine** parameter specifies the maximum number of asynchronous disk I/O requests that can be outstanding for a single engine at one time. See "max async i/os per server" on page 11-81 for more information.

max async i/os per server

Summary Information	
Name in pre-11.0 release	**cnmaxaio_server**
Default value	2147483647
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **max async i/os per server** parameter specifies the maximum number of asynchronous disk I/O requests that can be outstanding for Adaptive Server at one time. This limit is not affected by the number of online engines per Adaptive Server; **max async i/os per server** limits the total number of asynchronous I/Os a server can issue at one time, regardless of how many online engines it has. The configuration

parameter **max async i/os per engine** limits the number of outstanding I/Os per engine.

Most operating systems limit the number of asynchronous disk I/Os that can be processed at any one time; some operating systems limit the number of asynchronous I/Os per operating system process, some limit the number of asynchronous I/Os per system, and some do both. If an application exceeds these limits, the operating system returns an error message. Because operating system calls are relatively expensive, it is inefficient for Adaptive Server to attempt to perform asynchronous I/Os that get rejected by the operating system.

To avoid this, Adaptive Server maintains a count of the outstanding, asynchronous I/Os per engine and per server; if an engine issues an asynchronous I/O that would exceed either **max async i/os per engine** or **max async i/os per server**, Adaptive Server delays the I/O until enough outstanding I/Os have completed to fall below the limit that was exceeded.

For example, assume an operating system limit of 200 asynchronous I/Os per system and 75 per process and an Adaptive Server with three online engines. The engines currently have a total of 200 asynchronous I/Os pending, distributed according to the following table:

Engine	Number of I/Os Pending	Outcome
0	60	Engine 0 delays any further asynchronous I/Os until the total for the server is under the operating system **per-system** limit and then continues issuing asynchronous I/Os.
1	75	Engine 1 delays any further asynchronous I/Os until the per-engine total is under the operating system **per-process** limit and then continues issuing asynchronous I/Os.
2	65	Engine 2 delays any further asynchronous I/Os until the total for server is under the operating system **per-system** limit and then continues issuing asynchronous I/Os.

All I/Os (both asynchronous and synchronous) require a disk I/O structure, so the total number of outstanding disk I/Os is limited by the value of **disk i/o structures**. It is slightly more efficient for Adaptive Server to delay the I/O because it cannot get a disk I/O structure

than because the I/O request exceeds **max i/os per server**. You should set **max async i/os per server** equal to the value of **disk i/o structures.** See "disk i/o structures" on page 11-37.

If the limits for asynchronous I/O can be tuned on your operating system, make sure they are set high enough for Adaptive Server. There is no penalty for setting them as high as needed.

Use **sp_sysmon** to see if the per server or per engine limits are delaying I/O on your system. If **sp_sysmon** shows that Adaptive Server exceeded the limit for outstanding requests per engine or per server, raise the value of the corresponding parameter. See "Maximum Outstanding I/Os" on page 24-90 in the *Performance and Tuning Guide*.

o/s file descriptors

Summary Information	
Name in pre-11.0 release	N/A
Default value	0
Range of values	Site-specific
Status	Read-only
Display level	Comprehensive
Required role	System Administrator

The **o/s file descriptors** parameter indicates the maximum per-process number of file descriptors configured for your operating system. This parameter is read-only and cannot be configured through Adaptive Server.

Many operating systems allow you to configure the number of file descriptors available per process. See your operating system documentation for further information on this.

The number of file descriptors available for Adaptive Server connections, which will be less than the value of **o/s file descriptors**, is stored in the variable *@@max_connections*. For more information on the number of file descriptors available for connections, see "Upper Limit to the maximum number of user connections" on page 11-138.

shared memory starting address

Summary Information	
Name in pre-11.0 release	**mrstart**
Default value	0
Range of values	Platform-specific
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **shared memory starting address** parameter determines the virtual address at which Adaptive Server starts its shared memory region.

It is unlikely that you will ever have a need to reconfigure **shared memory starting address**. You should do so only after consulting with Sybase Technical Support.

Parallel Queries

The following parameters configure Adaptive Server for parallel query processing. When Adaptive Server is configured for parallel query processing, the optimizer evaluates each query to determine whether it is eligible for parallel execution.

To determine the best values to use for the configuration parameters, and to understand how these values affect the optimizer, see Chapter 13, "Introduction to Parallel Query Processing," and Chapter 14, "Parallel Query Optimization," in the *Performance and Tuning Guide*.

The **number of worker processes**, **max parallel degree**, and **max scan parallel degree** configuration parameters control parallel query processing at the server level. Using the **parallel_degree**, **process_limit_action**, and **scan_parallel_degree** options to the **set** command can limit parallel optimization at the session level, and using the **parallel** keyword of the **select** command can limit parallel optimization of specific queries. Figure 11-4 shows the precedence of the configuration parameters and session parameters.

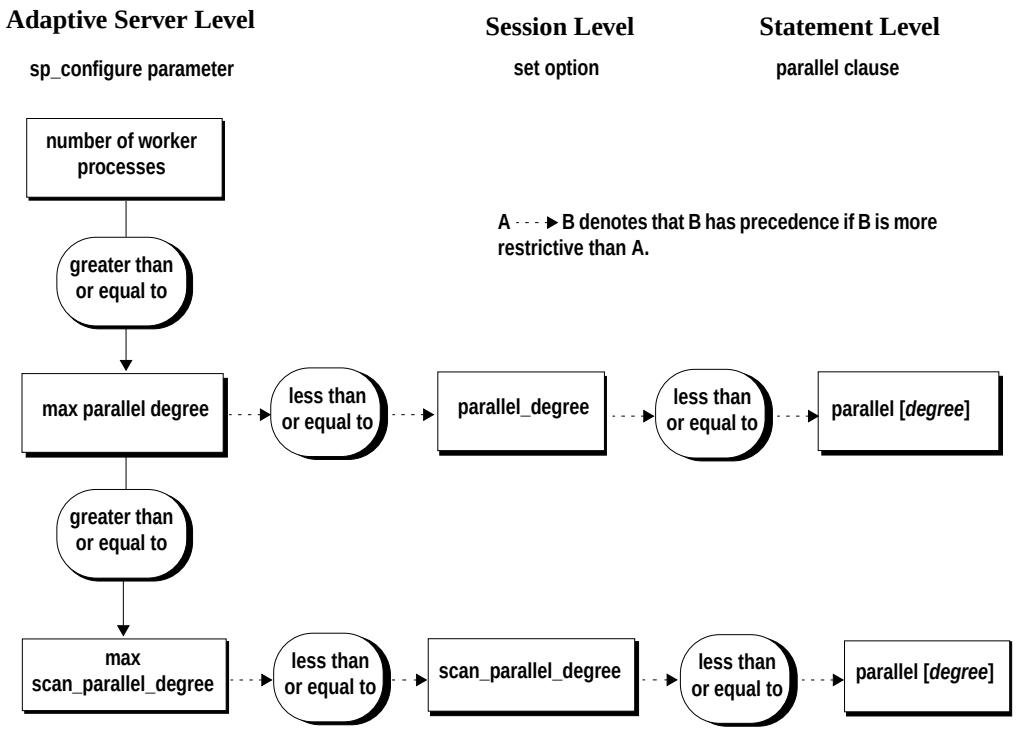


Figure 11-4: Precedence of parallel options

number of worker processes

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of worker processes** parameter specifies the maximum number of worker processes that Adaptive Server can use at any one time for all simultaneously running parallel queries combined.

Adaptive Server issues a warning message at start-up if there is insufficient memory to create the specified number of worker processes. The **memory per worker process** configuration parameter controls the memory allocated to each worker process.

max parallel degree

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	1–255
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **max parallel degree** parameter specifies the server-wide maximum number of worker processes allowed per table or index scan. This is called the **maximum degree of parallelism**.

This parameter limits the number of worker processes that a query may use. If this number is too low, the performance gain for a given query may not be as significant as it could be; if the number is too high, the server may compile plans that require more processes than

are actually available at execution time, or the system may become saturated, resulting in decreased throughput. To enable parallel partition scans, set this parameter to be equal to or greater than the number of partitions in the table you are querying. For

The value of this parameter must be less than or equal to the current value of **number of worker processes.**

If you set **max parallel degree** to 1, Adaptive Server scans all tables or indexes serially.

If you change this parameter, the new value affects subsequent operations only; Adaptive Server does not recompile query plans that are in the procedure cache. Plans already in the procedure cache continue to execute using the number of worker processes chosen by the optimizer when they were compiled. If you increase the value of **max parallel degree**, query plans in the procedure cache might be suboptimal until they are recompiled. If you decrease the value of **max parallel degree**, the plans currently in the procedure cache might use more processes, exceeding the current server-wide limit, which means that query plans in the cache might be suboptimal until they are recompiled.

For more information on parallel sorting, see Chapter 15, "Parallel Sorting," in the *Performance and Tuning Guide*.

max scan parallel degree

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	1–255
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **max scan parallel degree** parameter specifies the server-wide maximum degree of parallelism for hash-based scans. Hash-based scans may be used for the following access methods:

- Parallel index scans for partitioned and nonpartitioned tables
- Parallel table scans for nonpartitioned tables

The optimizer uses this parameter as a guideline when it selects the number of processes to use for parallel, nonpartition-based scan operations. It does not apply to parallel sort. Because there is no partitioning to spread the data across devices, parallel processes can be accessing the same device during the scan. This can cause additional disk contention and head movement, which can degrade performance. To prevent multiple disk accesses from becoming a problem, use this parameter to reduce the maximum number of processes that can access the table in parallel.

If this number is too low, the performance gain for a given query will not be as significant as it could be; if the number is too large, the server may compile plans that use enough processes to make disk access less efficient. A general rule of thumb is to set this parameter to no more than 2 or 3, because it takes only 2 to 3 worker processes to fully utilize the I/O of a given physical device.

Set the value of this parameter to less than or equal to the current value of **max parallel degree**. Adaptive Server returns an error if you specify a number larger than the **max parallel degree** value.

If you set **max scan parallel degree** to 1, Adaptive Server does not perform hash-based scans.

If you change this parameter, the new value will affect subsequent operations only. To recompile any query plans in the procedure cache, use **sp_recompile** *table_name*.

memory per worker process

Summary Information	
Name in pre-11.0 release	N/A
Default value	1024
Range of values	1024–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **memory per worker process** parameter specifies the amount of memory (in bytes) used by worker processes. Each worker process requires memory for messaging during query processing. This memory is allocated from a shared memory pool; this size of this

pool is **memory per worker process** times **number of worker processes**. For most query processing, the default size of 1024 is more than adequate. If you use **dbcc checkstorage**, and have **number of worker processes** set to 1, you may need to increase **memory per worker process** to 1792 bytes. See "Avg Mem Ever Used by a WP" on page 24-19 in the *Performance and Tuning Guide* for information on setting this parameter.

For more information on Adaptive Server's memory allocation, see Chapter 8, "Configuring Memory."

Physical Memory

The parameters in this group configure your machine's physical memory resources.

additional network memory

Summary Information	
Name in pre-11.0 release	**additional network memory**
Default value	0
Range of values	0–2147483647
Status	Static
Display level	Intermediate
Required role	System Administrator

The **additional network memory** parameter sets the maximum size of additional memory that can be used for network packets that are larger than the default packet size. Adaptive Server rounds down the value you enter to the nearest 2K value. The default value for **additional network memory** is 0, which means that no extra space is allocated for large packets.

If you increase **max network packet size** but do not increase **additional network memory**, clients cannot use packet sizes that are larger than the default size, because all allocated network memory is reserved for users at the default size. Adaptive Server guarantees that every user connection can log in at the default packet size. In this situation, users who request a large packet size when they log in receive a warning message telling them that their application will use the default size.

Increasing **additional network memory** may improve performance for applications that transfer large amounts of data. To determine the value for **additional network memory** when your applications use larger packet sizes:

- Estimate the number of simultaneous users who will request the large packet sizes, and the sizes their applications will request,
- Multiply this sum by 3, since each connection needs three buffers,
- Add 2 percent for overhead, and
- Round the value to the next highest multiple of 2048.

For example, if you estimate these simultaneous needs for larger packet sizes:

Application	Packet Size	Overhead
bcp	8192	
Client-Library	8192	
Client-Library	4096	
Client-Library	4096	
Total	24576	
Multiply by 3 buffers / user	*3	
	73728	
Compute 2% overhead		* .02=1474
Add overhead	+ 1474	
Additional network memory	75202	
Round up to multiple of 2048	75776	

You should set **additional network memory** to 75,776 bytes.

lock shared memory

Summary Information	
Name in pre-11.0 release	**T1611** (trace flag)
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **lock shared memory** parameter disallows swapping of Adaptive Server pages to disk and allows the operating system kernel to avoid the server's internal page locking code. This can improve performance by reducing disk reads, which are expensive.

Not all platforms support shared memory locking. Even if your platform does support it, the **lock shared memory** parameter may fail because of incorrectly set permissions, insufficient physical memory, or for other reasons. See the configuration documentation for your platform for information on specific requirements on shared memory locking.

max SQL text monitored

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **max SQL text monitored** parameter specifies the amount of memory allocated per user connection for saving SQL text to memory shared by Adaptive Server Monitor.

Initially, the amount of memory allocated for saving text is 0, and since this parameter is static, you must restart Adaptive Server before you can start saving SQL Text.

If you do not allocate enough memory for the batch statements, the text you want to view may be in the section of the batch that is truncated. Sybase recommends an initial value of 1024 bytes of memory per user connection. The maximum number of bytes saved for each client connection is between 0 (the default) and 2,147,483,647 (the theoretical limit).

The total memory allocated for the SQL Text from shared memory is the product of **max SQL text monitored** multiplied by the currently configured number of user connections.

For more information on **max SQL text monitored**, see "Configuring Adaptive Server to Save SQL Batch Text" on page 4-17.

total memory

Summary Information	
Name in pre-11.0 release	**memory**
Default value	Platform-specific
Range of values	Platform-specific minimum–2147483647
Status	Static
Display level	Intermediate
Required role	System Administrator

The **total memory** parameter sets the size of memory, in 2K units, that Adaptive Server allocates from the operating system. The default value of **total memory** varies from platform to platform. See the configuration documentation for your platform for the value on your operating system.

➤ ***Note***

The "Memory Used" column in the output reports this size (and all memory use) in 1K, not 2K units.

The more memory that is available, the more resources Adaptive Server has for internal buffers and caches, reducing the number of

times the server has to read data from disk for static information or compiled procedure plans. There is no performance penalty for configuring Adaptive Server to use the maximum memory available to it on your computer. However, be sure to assess the other memory needs on your system, or Adaptive Server may not be able to acquire enough memory to start. See Chapter 8, "Configuring Memory," for instructions on how to maximize the amount of **total memory** for Adaptive Server.

If Adaptive Server Cannot Start

When Adaptive Server starts, it must acquire the full amount of memory, set by **total memory**, from the operating system. If Adaptive Server does not start because this amount of memory is not available to it, reduce the memory requirements for Adaptive Server by changing the value of the **total memory** parameter in the server's configuration file. You may also need to reduce the values for other configuration parameters that require large amounts of memory. Then restart Adaptive Server to use the memory needed by the new values. If Adaptive Server fails to start because the total of other configuration parameter values is higher than the **total memory** value, see Chapter 8, "Configuring Memory," for information about configuration parameters that use memory.

Processors

The parameters in this group configure processors in an SMP environment.

max online engines

Summary Information	
Name in pre-11.0 release	**max online engines**
Default value	1
Range of values	1–32
Status	Static
Display level	Intermediate
Required role	System Administrator

The **max engines online** parameter specifies the maximum number of Adaptive Server engines that can be online at any one time in an SMP environment. See Chapter 10, "Managing Multiprocessor Servers," for a detailed discussion of how to set this parameter for your SMP environment.

At start-up, Adaptive Server starts with a single engine and completes its initialization, including recovery of all databases. Its final task is to allocate additional server engines. Each engine accesses common data structures in shared memory.

When tuning the **max engines online** parameter:

- Never have more online engines than there are CPUs.
- Depending on overall system load (including applications other than Adaptive Server), you may achieve optimal throughput by leaving some CPUs free to run non-Adaptive Server processes.
- Better throughput can be achieved by running fewer engines with high CPU use, rather than by running more engines with low CPU use.
- Scalability is application-dependent. You should conduct extensive benchmarks on your application to determine the best configuration of online engines.

See "Kernel Utilization" on page 24-11 of the *Performance and Tuning Guide* for more information on performance and engine tuning.

Rep Agent Thread Administration

The parameter in this group configures replication via Replication Server®.

enable rep agent threads

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Basic
Required role	System Administrator

For users of Replication Server, the **enable rep agent threads** parameter enables the RepAgent thread within Adaptive Server.

Through release 11.0.3 of Replication Server, the Log Transfer Manager (LTM), a replication system component, transfers replication data to the Replication Server. Beginning with Replication Server releases after 11.0.3, transfer of replication data will be handled by the RepAgent, which will run as a thread under Adaptive Server. Setting the **enable rep agent threads** parameter enables the use of this feature.

Other steps are also required to enable replication. For more information, see the Replication Server documentation.

SQL Server Administration

The parameters in this group are related to general Adaptive Server administration.

allow backward scans

Summary Information	
Name in pre-11.0 release	N / A
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **allow backward scans** parameter controls how the optimizer performs **select** queries that contain the **order by...desc** command:

- When the value is set to 1, the optimizer can access the index or table rows by following the page chain in descending index order.
- When the value is set to 0, the optimizer selects the rows into a worktable by following the index page pointers in ascending order and then sorts the worktable in descending order.

The first method—performing backward scans—can speed access to tables that need results ordered by descending column values. Some applications, however, may experience deadlocks due to backward scans. In particular, look for increased deadlocking if you have **delete** or **update** queries that scan forward using the same index. There may also be deadlocks due to page splits in the index.

You can use the **print deadlock information** parameter to send messages about deadlocks to the error log. See “print deadlock information” on page 11-122. Alternatively, you can use the system procedure **sp_sysmon** to check for deadlocking. See “Deadlocks by Lock Type” on page 24-64 in the *Performance and Tuning Guide* for more information on deadlocks.

allow nested triggers

Summary Information	
Name in pre-11.0 release	**nested trigger**
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Intermediate
Required role	System Administrator

The **allow nested triggers** parameter controls the use of nested triggers. When the value is set to 1, data modifications made by triggers can fire other triggers. Set **allow nested triggers** to 0 to disable nested triggers. The default is 1. A **set** option, **self_recursion**, controls whether the modifications made by a trigger can cause that trigger to fire again.

allow resource limits

Summary Information	
Name in pre-11.0 release	N/A
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **allow resource limits** parameter controls the use of resource limits. When the value is set to 1, the server allocates internal memory for time ranges, resource limits, and internal server alarms. The server also internally assigns applicable ranges and limits to user sessions. The output of **sp_configure** displays the optimizer's cost estimate for a query. Set **allow resource limits** to 0 to disable resource limits. The default is 0.

allow updates to system tables

Summary Information	
Name in pre-11.0 release	**allow updates**
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **allow updates to system tables** parameter enables users with the System Administrator role to make changes to the system tables and to create stored procedures that can modify system tables. A database administrator can update system tables in any that he or she owns if **allow updates to system tables** is enabled.

System tables include:

- All Sybase-supplied tables in the *master* database
- All tables in user databases that begin with "sys" and that have an ID value in the *sysobjects* table of less than or equal to 100

◆ ***WARNING!***

Incorrect alteration of a system table can result in database corruption and loss of data. Always use begin transaction when modifying a system table to protect against errors that could corrupt your databases. Immediately after finishing your modifications, disable allow updates to system tables; while it is enabled, any user with the System Administrator role can modify system tables or create stored procedures that can modify system tables.

Stored procedures and triggers you create while the **allow updates to system tables** parameter is set **on** are always able to update the system tables, even after the parameter has been set **off**. Whenever you set **allow updates to system tables** to **on**, you are creating a "window of vulnerability," a period of time during which Adaptive Server users can alter the system tables or create a stored procedure with which the system tables can be altered in the future.

Because the system tables are so critical, it is best to set this parameter to **on** only in highly controlled situations. If you want to guarantee that no other users can access Adaptive Server while the system tables can be directly updated, you can restart Adaptive Server in single-user mode. For details, see **startserver** and **dataserver** in the *Utility Programs* manual for your platform manual.

cpu accounting flush interval

Summary Information	
Name in pre-11.0 release	**cpu flush**
Default value	200
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **cpu accounting flush interval** parameter specifies the amount of time, in machine clock ticks, that Adaptive Server waits before flushing CPU usage statistics for each user from *sysprocesses* to *syslogins*, a procedure used in chargeback accounting. (Note that this is measured in **machine** clock ticks, not Adaptive Server clock ticks.)

When a user logs into Adaptive Server, the server begins accumulating figures for CPU usage for that user process in *sysprocesses*. When a user logs off Adaptive Server, or when the value of **cpu accounting flush interval** is exceeded, the accumulated CPU usage statistics are flushed from *sysprocesses* to *syslogins*. These statistics continue accumulating in *syslogins* until you clear the totals by using **sp_clearstats**. You can display the current totals from *syslogins* by using **sp_reportstats**.

The value to which you set **cpu accounting flush interval** depends on the type of reporting you intend to do. If you intend to run reports on a monthly basis, setting **cpu accounting flush interval** to a relatively high value makes sense. With infrequent reporting, it is less critical that the data in *syslogins* be updated as often.

On the other hand, if you intend to do periodic ad hoc selects on the *totcpu* column in *syslogins*, to determine CPU usage by process, it makes sense to set **cpu accounting flush interval** to a lower value. Doing so

increases the likelihood of the data in *syslogins* being up to date when you execute your selects.

Setting **cpu accounting flush interval** to a low value may cause processes to be mistakenly identified as potential deadlock victims by the lock manager. When the lock manager detects a deadlock, it checks the amount of CPU time accumulated by each of the competing processes. The process with the lesser amount is chosen as the deadlock victim and is terminated by the lock manager. When **cpu accounting flush interval** is set to a low value, the task handlers that store CPU usage information for processes are initialized more frequently, thus making processes appear as if they have accumulated less CPU time than they actually have. Because of this, the lock manager may select a process as the deadlock victim when, in fact, that process has more accumulated CPU time than the competing process.

If you do not intend to report on CPU usage at all, set **cpu accounting flush interval** to its maximum value. This reduces the number of times *syslogins* is updated and reduces the number of times its pages need to be written to disk.

cpu grace time

Summary Information	
Name in pre-11.0 release	**ctimemax**
Default value	500
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **cpu grace time** parameter, together with the **time slice** parameter, specify the maximum amount of time (in clock ticks) that a user process can run without yielding the CPU before Adaptive Server preempts it and terminates it with a time-slice error.

When a process exceeds **cpu grace time** Adaptive Server “infects” it by removing the process from the internal queues. The process is killed, but Adaptive Server is not affected. This prevents runaway processes from monopolizing the CPU. If any of your user processes become infected, you may be able to temporarily fix the problem by increasing the value of **cpu grace time**. However, you must be sure that

the problem really is a process that takes more than the current value of **cpu grace time** to complete, rather than a runaway process.

Temporarily increasing the **cpu grace time** value should be considered a workaround, not a permanent fix, since it may cause other complications; see "time slice" on page 11-127. Also, see Chapter 21, "How Adaptive Server Uses Engines and CPUs," and "Adaptive Server Execution Task Scheduling" on page 21-7 of the *Performance and Tuning Guide* for a more detailed discussion of task scheduling.

default database size

Summary Information	
Name in pre-11.0 release	**database size**
Default value	2
Range of values	2–10000
Status	Static
Display level	Intermediate
Required role	System Administrator

The **default database size** parameter sets the default number of megabytes allocated to a new user database if the **create database** statement is issued without any size parameters. A database size given in a **create database** statement takes precedence over the value set by this configuration parameter.

The default run value is 2MB. If most of the new databases on your Adaptive Server require more than 2MB, you may want to increase this value.

➤ ***Note***

You must increase the default **database size** value if you alter the *model* database to make it larger than 2MB, because the **create database** command copies *model* to create a new user database.

default fill factor percent

Summary Information	
Name in pre-11.0 release	**fillfactor**
Default value	0
Range of values	0–100
Status	Static
Display level	Intermediate
Required role	System Administrator

The **default fill factor percent** parameter determines how full Adaptive Server makes each index page when it is creating a new index on existing data, unless the fill factor is specified in the **create index** statement. The **fillfactor** percentage is relevant only at the time the index is created. As the data changes, the pages are not maintained at any particular level of fullness.

The **default fill factor percent** parameter affects:

- The amount of storage space used by your data

  Adaptive Server redistributes the data as it creates the clustered index.

- Performance

  Splitting up pages uses Adaptive Server resources.

There is seldom a reason to change the **default fill factor percent** parameter, especially since you can override it in the **create index** command. For more information about the fill factor percentage, see **create index** in the *Adaptive Server Reference Manual.*

dump on conditions

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **dump on conditions** parameter determines whether Adaptive Server generates a dump of data in shared memory when it encounters the conditions specified in the **maximum dump conditions** parameter.

➤ ***Note***

The **dump on conditions** parameter is included for use by Sybase Technical Support only. Do not modify it unless you are instructed to do so by Sybase Technical Support.

event buffers per engine

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **event buffers per engine** parameter specifies the number of events per Adaptive Server engine that can be monitored simultaneously by Adaptive Server Monitor. Events are used by Adaptive Server

Monitor for observing Adaptive Server performance; if you are not using Adaptive Server Monitor, set this parameter to 1.

The value to which you set **event buffers per engine** depends on the number of engines in your configuration, the level of activity on your Adaptive Server, and the kinds of applications you are running.

Setting **event buffers per engine** to a low value may result in the loss of event information. The default value, 100, is likely to be too low for most sites. Values of 2000 and above may be more reasonable for general monitoring. However, you need to experiment in order to determine the appropriate value for your site.

In general, setting **event buffers per engine** to a high value may reduce the amount of performance degradation that Adaptive Server Monitor causes Adaptive Server.

Each event buffer uses 100 bytes of memory. To determine the total amount of memory used by a particular value for **event buffers per engine**, multiply the value by the number of Adaptive Server engines in your configuration.

housekeeper free write percent

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **housekeeper free write percent** parameter specifies the maximum percentage by which the housekeeper task can increase database writes. Valid values range from 0 to 100.

For example, to stop the housekeeper task from working when the frequency of database writes reaches 5 percent above normal, set **housekeeper free write percent** to 5:

```
sp_configure "housekeeper free write percent", 5
```

When Adaptive Server has no user tasks to process, the housekeeper task automatically begins writing changed pages from cache to disk.

These writes result in improved CPU utilization, decreased need for buffer washing during transaction processing, and shorter checkpoints.

In applications that repeatedly update the same database page, the housekeeper may initiate some unnecessary database writes. Although these writes occur only during the server's idle cycles, they may be unacceptable on systems with overloaded disks.

By default, the **housekeeper free write percent** parameter is set to 1. This allows the housekeeper task to increase disk I/O by a maximum of 1 percent. This results in improved performance and recovery speed on most systems.

To disable the housekeeper task, set the value of **housekeeper free write percent** to 0:

```
sp_configure "housekeeper free write percent", 0
```

To allow the housekeeper task to work continuously, regardless of the percentage of additional database writes, set **housekeeper free write percent** to 100:

```
sp_configure "housekeeper free write percent", 100
```

Use **sp_sysmon** to monitor housekeeper performance. See "Pool Turnover" on page 24-78 and "Increasing the Housekeeper Batch Limit" on page 24-87 in the *Performance and Tuning Guide.*

It might also be helpful to look at the number of free checkpoints initiated by the housekeeper task. The *Performance and Tuning Guide* describes this output in "Number of Free Checkpoints" on page 24-87.

identity burning set factor

Summary Information	
Name in pre-11.0 release	**identity burning set factor**
Default value	5000
Range of values	1–9999999
Status	Static
Display level	Intermediate
Required role	System Administrator

IDENTITY columns are columns of type *numeric* and scale zero whose values are generated by Adaptive Server. Column values can range from a low of 1 to a high determined by the column precision.

For each table with an IDENTITY column, Adaptive Server divides the set of possible column values into blocks of consecutive numbers, and makes one block at a time available in memory. Each time you insert a row into a table, Adaptive Server assigns the IDENTITY column the next available value from the block. When all the numbers in a block have been used, the next block becomes available.

This method of choosing IDENTITY column values improves server performance. When Adaptive Server assigns a new column value, it reads the current maximum value from memory and adds 1. Disk access becomes necessary only after all values within the block have been used. Because all remaining numbers in a block are discarded in the event of server failure (or **shutdown with nowait**), this method can lead to gaps in IDENTITY column values.

Use **identity burning set factor** to change the percentage of potential column values that is made available in each block. This number should be high enough for good performance, but not so high that gaps in column values are unacceptably large. The default value, 5000, releases .05 percent of the potential IDENTITY column values for use at one time.

To get the correct value for **sp_configure**, express the percentage in decimal form, and then multiply it by 10^7 (10,000,000). For example, to release 15 percent (.15) of the potential IDENTITY column values at a time, specify a value of .15 times 10^7 (or 1,500,000) in **sp_configure**:

```
sp_configure "identity burning set factor", 1500000
```

identity grab size

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	1–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **identity grab size** parameter allows each Adaptive Server process to reserve a block of IDENTITY column values for inserts into tables that have an IDENTITY column.

This is useful if you are doing inserts, and you want all the inserted data to have contiguous IDENTITY numbers. For instance, if you are entering payroll data, and you want all records associated with a particular department to be located within the same block of rows, set **identity grab size** to the number of records for that department.

The **identity grab size** parameter applies to all users on Adaptive Server. Large **identity grab size** values result in large gaps in the IDENTITY column when many users insert data into tables that have IDENTITY columns.

Sybase recommends setting **identity grab size** to a value that is large enough to accommodate the largest group of records you want to insert into contiguous rows.

i/o accounting flush interval

Summary Information	
Name in pre-11.0 release	**i/o flush**
Default value	1000
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **i/o accounting flush interval** parameter specifies the amount of time, in machine clock ticks, that Adaptive Server waits before flushing I/O statistics for each user from *sysprocesses* to *syslogins*. (Note that this is measured in **machine** clock ticks, not Adaptive Server clock ticks.) This is used for chargeback accounting.

When a user logs into Adaptive Server, the server begins accumulating I/O statistics for that user process in *sysprocesses*. When the value of **i/o accounting statistics interval** is exceeded, or a user logs off Adaptive Server, the accumulated I/O statistics for that user are flushed from *sysprocesses* to *syslogins*. These statistics continue accumulating in *syslogins* until you clear the totals by using **sp_clearstats**. You display the current totals from *syslogins* by using **sp_reportstats**.

The value to which you set **i/o accounting flush interval** depends on the type of reporting you intend to do. If you intend to run reports on a monthly basis, setting **i/o accounting flush interval** to a relatively high value makes sense. This is because, with infrequent reporting, it is less critical that the data in *syslogins* be updated frequently.

If you intend to do periodic ad hoc selects on the *totio* column *syslogins*, to determine I/O volume by process, it makes sense to set **i/o accounting flush interval** to a lower value. Doing so increases the likelihood of the data in *syslogins* being up to date when you execute your selects.

If you do not intend to report on I/O statistics at all, set **i/o accounting flush interval** to its maximum value. This reduces the number of times *syslogins* is updated and the number of times its pages need to be written to disk.

i/o polling process count

Summary Information	
Name in pre-11.0 release	**cmaxscheds**
Default value	10
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **i/o polling process count** parameter specifies the maximum number of processes that can be run by Adaptive Server before the scheduler checks for disk and/or network I/O completions. Tuning **i/o polling process count** affects both the response time and the throughput of Adaptive Server.

Adaptive Server checks for disk or network I/O completions:

- If the number of tasks run since the last time Adaptive Server checked for I/O completions equals the value for **i/o polling process count**, and
- At every Adaptive Server clock tick.

As a general rule, increasing the value of **i/o polling process count** may increase throughput for applications that generate a lot of disk and network I/O. Conversely, decreasing the value may improve process response time in these applications, possibly at the risk of lowering throughput.

If your applications create both I/O-bound tasks and CPU-bound tasks, tuning **i/o polling process count** to a low value (1–2) ensures that I/O-bound tasks get access to CPU cycles.

For OLTP applications (or any I/O-bound application with a lot of user connections and short transactions), tuning **i/o polling process count** to a value in the range of 20–30 may increase throughput, but it may also increase response time.

When tuning **i/o polling process count**, consider three other parameters:

- **sql server clock tick length**, which specifies the duration of Adaptive Server's clock tick in microseconds, and thus affects the amount of time the server will wait before checking for outstanding I/O. See "sql server clock tick length" on page 11-126.

- **time slice**, which specifies the number of clock ticks Adaptive Server's scheduler allows a user process to run. This parameter is useful for determining how long CPU-bound tasks run before yielding the CPU. See "time slice" on page 11-127.
- **cpu grace time**, which specifies the maximum amount of time (in clock ticks) a user process can run without yielding the CPU before Adaptive Server preempts it and terminates it with a time-slice error. See "cpu grace time" on page 11-100.

Use **sp_sysmon** to determine the effect of changing the **i/o polling process count** parameter. See "Average Network I/Os per Check" and "Total Disk I/O Checks" on page 24-16 in the *Performance and Tuning Guide*.

lock promotion HWM

Summary Information	
Name in pre-11.0 release	N/A
Default value	200
Range of values	2–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **lock promotion HWM** (high water mark) parameter, together with **lock promotion LWM** (low water mark) and **lock promotion PCT** (percentage), specify the number of page locks permitted during a single scan session of a table or an index before Adaptive Server attempts to escalate from page locks to a table lock.

The **lock promotion HWM** parameter sets a maximum number of locks allowed on a table before Adaptive Server escalates to a table lock. When the number of locks acquired during a scan session exceeds **lock promotion HWM**, Adaptive Server attempts to acquire a table lock. The **lock promotion HWM** value cannot be higher than **number of locks** value.

For more detailed information on scan sessions and setting up lock promotion limits, see "Configuring Locks and Lock Promotion Thresholds" on page 5-46 in the *Performance and Tuning Guide*.

The default value (200) for **lock promotion HWM** is appropriate for most applications. You might want to raise the value to avoid table

locking. For example, if you know that there are regular updates to 500 rows on a table with thousands of rows, you can increase concurrency for the tables by setting **lock promotion HWM** to around 500.

You can also configure lock promotion at the per-object level. See **sp_setpglockpromote** in the *Adaptive Server Reference Manual*.

Use **sp_sysmon** to see how changing the **lock promotion HWM** parameter affects the number of lock promotions. **sp_sysmon** reports the number of exclusive page to exclusive table lock promotions and shared page to shared table lock promotions. See "Lock Promotions" on page 24-66 in the *Performance and Tuning Guide*.

lock promotion LWM

Summary Information	
Name in pre-11.0 release	N / A
Default value	200
Range of values	2–value of **lock promotion HWM**
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **lock promotion LWM** (low water mark) parameter, together with the **lock promotion HWM** (high water mark) and **lock promotion PCT** (percentage), specify the number of page locks permitted during a single scan session of a table or an index before Adaptive Server attempts to promote from page locks to a table lock.

The **lock promotion LWM** sets the number of locks below which Adaptive Server does not attempt to issue a table lock on the object. The **lock promotion LWM** parameter must be less than or equal to **lock promotion HWM**.

For more detailed information on scan sessions and setting up lock promotion limits, see "Configuring Locks and Lock Promotion Thresholds" on page 5-46 in the *Performance and Tuning Guide*.

The default value (200) for **lock promotion LWM** is sufficient for most applications. If Adaptive Server runs out of locks (except for an isolated incident), you should increase **number of locks.**

You can also configure lock promotion at the per-object level. See **sp_setpglockpromote** in the *Adaptive Server Reference Manual*.

lock promotion PCT

Summary Information	
Name in pre-11.0 release	N / A
Default value	100
Range of values	1–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

If the number of locks is between **lock promotion LWM** (low water mark) and **lock promotion HWM** (high water mark), the **lock promotion PCT** (percentage) parameter sets the percentage of page locks (based on the table size) above which Adaptive Server attempts to acquire a table lock when the number of locks is between **lock promotion HWM** and **lock promotion LWM**.

For more detailed information on setting up lock promotion limits, see “Configuring Locks and Lock Promotion Thresholds” on page 5-46 in the *Performance and Tuning Guide*.

The default value (100) for **lock promotion PCT** is appropriate for most applications.

You can also configure lock promotion at the per-object level. See **sp_setpglockpromote** in the *Adaptive Server Reference Manual*.

maximum dump conditions

Summary Information	
Name in pre-11.0 release	N / A
Default value	10
Range of values	10–100
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **maximum dump conditions** parameter sets the maximum number of conditions you can specify under which Adaptive Server generates a dump of data in shared memory.

➤ ***Note***

The **maximum dump conditions** parameter is included for use by Sybase Technical Support only. Do not modify it unless you are instructed to do so by Sybase Technical Support.

number of alarms

Summary Information	
Name in pre-11.0 release	**cnalarm**
Default value	40
Range of values	5–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of alarms** parameter specifies the number of alarm structures allocated by Adaptive Server.

The Transact-SQL command **waitfor** defines a specific time, time interval, or event for the execution of a statement block, stored

procedure, or transaction. Adaptive Server uses alarms to execute **waitfor** commands correctly.

The number of alarms you need is determined by the needs of your applications and the number of simultaneous instances of each application that uses **waitfor**. When Adaptive Server needs more alarms than are currently allocated, the following message is written to the error log:

```
uasetalarm: no more alarms available
```

Each alarm structure uses 20 bytes of memory. If you raise the **number of alarms value** significantly, you should adjust **total memory** accordingly.

number of aux scan descriptors

Summary Information	
Name in pre-11.0 release	N / A
Default value	200
Range of values	0–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of aux scan descriptors** parameter sets the number of auxiliary scan descriptors needed for a server.

When a table has references to other tables, Adaptive Server scans the referenced tables when the table is being modified. Adaptive Server uses **scan descriptors** to manage the table scans.

Each Adaptive Server connection is preallocated with 28 scan descriptors. In addition to this number, there is a global pool of auxiliary scan descriptors that allow connections to allocate and free scan descriptors dynamically, as needed. For tables with a large number of references, a connection may need auxiliary scan descriptors after it has used its allotted number of scan descriptors. When this happens, Adaptive Server displays a warning message, saying that the query requires more auxiliary scan descriptors, and then rolls back the user transaction in that connection.

If none of the tables in your databases uses referential constraints, set **number of aux scan descriptors** to 0. If the tables in your databases do use referential integrity, and a connection needs more scan descriptors to

run a referential integrity query, use either of the following methods to remedy the problem:

- Redesign the table's schema so that it uses fewer scan descriptors. You can find how many auxiliary scan descriptors a query will use by using **set showplan, noexec on** before running the query.
- Increase the **number of aux scan descriptors** setting, using the procedures in the following section.

Estimating and Configuring Auxiliary Scan Descriptors

First, estimate the number of scan descriptors needed:

1. Use the system procedure **sp_helpconstraint** to find the largest number of references used in a single table in the current database.

   Running **sp_helpconstraint** by itself generates a report listing the number of references used by each table in the database, in descending order. For example:

   ```
   sp_helpconstraint
   ```

   ```
   id           name                    Num_referential_constraints
   -----------  ----------------------  ---------------------------
      48003202  west_sales                                       38
      11200343  new_discounts                                    10
      16003088  royalties                                         2
   ```

   In this example, the *west_sales* table has the most referential constraints in the current database.

2. Estimate the number of simultaneous queries against the table with the highest number of referential constraints.

   You can base this value on the number of users who have access to the server. The value does not have to be exact, but it should be close.

3. Calculate the number of needed auxiliary scan descriptors, based on the value estimated in step 2.

   For example, the *west_sales* table has 38 references; suppose that up to 15 queries are run simultaneously on *west_sales*. You need a total of (38 – 28) * 15 = 150 auxiliary scan descriptors for the database in which *west_sales* resides, because 28 is the number of scan descriptors preallocated to each connection.

4. Repeat steps 1–3 for each database that uses referential constraints in its tables.

Include the system databases such as *model* and *tempdb,* unless you are sure that no referential constraints have been added to these databases.

Next, configure Adaptive Server to use the estimated number of needed scan descriptors:

1. Use the **number of aux scan descriptors** configuration parameter to reconfigure Adaptive Server to an appropriate value.

   For example, suppose you need the following auxiliary scan descriptors for a database:

   ```
   sales.west_sales      150
   employees.cal_emp     200
   products.orders       150
                         ---
                         500
   ```

   You would configure the server as follows:

```
sp_configure "number of aux scan descriptors", 500
```

2. Restart Adaptive Server so that the new setting takes effect.
3. If there is a lot of activity on the server, run **sp_monitorconfig** periodically at peak periods, as described in the following section.

Monitoring Scan Descriptor Usage

The setting in the preceding example (see step 1) is a good basis on which to start. Afterward, if there is a lot of activity on the server, run the system procedure **sp_monitorconfig** periodically, at peak periods, to monitor Adaptive Server. **sp_monitorconfig** displays the number of unused (free) scan descriptors, the number of auxiliary scan descriptors being used now, the percentage that is active, and the maximum number of scan descriptors used since the server was last started. **sp_monitorconfig** takes into consideration referential integrity queries running in parallel.

For example, suppose you configured the server for 500 auxiliary scan descriptors. After running tests on the tables used for referential integrity-intensive applications, suppose **sp_monitorconfig** returns the following output:

```
sp_monitorconfig "aux scan descriptors"
Usage information at date and time: Jan 24 1997 9:54AM.
```

```
Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of aux   260       240      48.00      380               NA
scan
descriptors
```

Of the 500 auxiliary scan descriptors configured, only 240 are being used, leaving 260 free. However, the maximum number of scan descriptors used since the last time Adaptive Server was started is 380. Therefore, the optimal number of auxiliary scan descriptors you should configure for the server is 400, to accommodate the 380 auxiliary scan descriptors, plus additional space for 20 more scan descriptors.

number of mailboxes

Summary Information	
Name in pre-11.0 release	**cnmbox**
Default value	30
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of mailboxes** parameter specifies the number of mailbox structures allocated by Adaptive Server. Mailboxes, which are used in conjunction with messages, are used internally by Adaptive Server for communication and synchronization between kernel service processes. Mailboxes are not used by user processes. Do not modify this parameter unless instructed to do so by Sybase Technical Support.

number of messages

Summary Information	
Name in pre-11.0 release	**cnmsg**
Default value	64
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of messages** parameter specifies the number of message structures allocated by Adaptive Server. Messages, which are used in conjunction with mailboxes, are used internally by Adaptive Server for communication and synchronization between kernel service processes. Messages are also used for coordination between a family of processes in parallel processing. Do not modify this parameter unless instructed to do so by Sybase Technical Support.

number of pre-allocated extents

Summary Information	
Name in pre-11.0 release	**cpreallocext**
Default value	2
Range of values	0–31
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **number of pre-allocated extents** parameter specifies the number of extents (eight pages) allocated in a single trip to the page manager. Currently it is only used by **bcp** to improve performance when copying in large amounts of data. By default, **bcp** allocates two extents at a time and writes an allocation record to the log each time.

Setting **number of pre-allocated extents** means that **bcp** allocates the specified number of extents each time it requires more space, and writes a single log record for the event. Setting the value to 0 disables

extent allocation so that a single page is allocated each time bulk copy needs a page. Since each page allocation is logged, this can greatly increase the amount of transaction log space required.

An object may be allocated more pages than actually needed, so the value of **number of pre-allocated extents** should be low if you are using **bcp** for small batches. If you are using **bcp** for large batches, increase the value of **number of pre-allocated extents** to reduce the amount of overhead required to allocate pages and to reduce the number of log records.

number of sort buffers

Summary Information	
Name in pre-11.0 release	**csortbufsize**
Default value	500
Range of values	0–32767
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **number of sort buffers** parameter specifies the number of 2K buffers used to hold pages read from input tables and perform index merges during sorts.

Sybase recommends that you leave the **number of sort buffers** parameter set to the default of 500 except when creating indexes in parallel. Setting the value too high can rob non-sorting processes of access to the 2K buffer pool in caches being used to perform sorts.

For more information on configuring this value for parallel **create index** statements, see “Configuring the number of sort buffers Parameter” on page 15-13 in the *Performance and Tuning Guide.*

partition groups

Summary Information	
Name in pre-11.0 release	N / A
Default value	1024
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **partition groups** parameter specifies how many **partition groups** are allocated for Adaptive Server. Partition groups are internal structures used by Adaptive Server to control access to individual partitions of a table.

A partition group is composed of 16 partition caches, each of which stores information about a single partition. All caches in a partition group are used to store information about the same partitioned table. If a table has fewer than 16 partitions, the unused partition caches are wasted. If a table has more than 16 partitions, it requires multiple partition groups.

The default value for the **partition groups** parameter is 1024. This allows a maximum of 1024 open partition groups and 16384 (1024 times 16) open partitions.

Adaptive Server allocates partition groups to a table when you partition the table or when you access it for the first time after setting this parameter before restarting Adaptive Server. If there are not enough partition groups for the table, you will not be able to access or partition the table.

partition spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	10
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the **partition spinlock ratio** parameter sets the number of rows in the internal partition caches that are protected by one **spinlock**.

Adaptive Server manages access to table partitions using internal **partition groups,** each of which contains partition caches. Each partition cache stores information about a partition (for example, the last page of the partition) that processes must use when accessing that partition.

By default, Adaptive Server systems are configured with **partition spinlock ratio** set to 10, or 1 spinlock for every 10 partition caches. Decreasing the value of the **partition spinlock ratio** parameter may have little impact on the performance of Adaptive Server. The default setting (10) for this parameter is correct for most servers.

For more information about configuring spinlock ratios, see “Configuring Spinlock Ratio Parameters” on page 10-9.

print deadlock information

Summary Information	
Name in pre-11.0 release	**T1204** (trace flag)
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **print deadlock information** parameter enables the printing of deadlock information to the error log.

If you are experiencing recurring deadlocks, setting **print deadlock information** to 1 provides you with information that can be useful in tracing the cause of the deadlocks.

➤ ***Note***

Setting **print deadlock information** to 1can seriously degrade Adaptive Server performance. For this reason, you should use it only when you are trying to determine the cause of deadlocks.

Use **sp_sysmon** output to determine whether deadlocks are occurring in your application. If they are, set **print deadlock information** parameter to 1 to learn more about why they are occurring. See “Deadlocks by Lock Type” on page 24-64 in the *Performance and Tuning Guide.*

runnable process search count

Summary Information	
Name in pre-11.0 release	**cschedspins**
Default value	2000
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **runnable process search count** parameter specifies the number of times an engine loops while looking for a runnable task before relinquishing the CPU to the operating system.

Adaptive Server engines check the run queue for runnable tasks whenever a task completes or exceeds its allotted time on the engine. At times, there will not be any tasks in the run queues. An engine can either relinquish the CPU to the operating system or continue to check for a task to run. Setting **runnable process search count** higher causes the engine to loop more times, thus holding the CPU for a longer time. Setting the **runnable process search count** lower causes the engine to release the CPU sooner.

If your machine is a uniprocessor that depends on helper threads to perform I/O, you may see some performance benefit from setting **runnable process search count** lower. The operating system may require more access to the CPU in order to perform network I/O, disk I/O, or other operating system tasks. If a client, such as a bulk copy operation, is running on the same machine as a single CPU server that uses helper threads, it can be especially important to allow both the server and the client access to the CPU.

For Adaptive Servers running on uniprocessor machines that do not use helper threads, and for multiprocessor machines, the default value provides good performance.

Use **sp_sysmon** to determine how the **runnable process search count** parameter affects Adaptive Server's use of CPU cycles, engine yields to the operating system, and blocking network checks. See "Engine Busy Utilization" on page 24-12, "CPU Yields by Engine" on page 24-14, and "Network Checks" on page 24-14 in the *Performance and Tuning Guide*.

size of auto identity column

Summary Information	
Name in pre-11.0 release	N / A
Default value	10
Range of values	1–38
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The **size of auto identity column** parameter sets the precision of IDENTITY columns that are automatically created with the **sp_dboption "auto identity"** and **"unique auto_identity index"** options.

The maximum value that can be inserted into an IDENTITY column is 10 PRECISION -1. Once an IDENTITY column reaches its maximum value, all further **insert** statements return an error that aborts the current transaction.

If you reach the maximum value of an IDENTITY column, use the **create table** command to create a table that is identical to the old one, but with a larger precision for the IDENTITY column. Once you have created the new table, use the **insert** command or **bcp** to copy data from the old table to the new one.

SQL Perfmon Integration (Windows NT only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Intermediate
Required role	System Administrator

The **SQL Perfmon Integration** parameter enables and disables the ability to monitor Adaptive Server statistics from the Windows NT Performance Monitor.

A value of 1 enables Adaptive Server performance monitoring in the Windows NT Performance Monitor. A value of 0 disables it.

Adaptive Server must be registered as an NT Service to support monitor integration. This occurs automatically when:

- You start Adaptive Server using the Services Manager in the Sybase for Windows NT program group
- You use the Services option in the Control Panel
- You have configured Windows NT to start Adaptive Server as an automatic service.

See *Configuring Adaptive Server for Windows NT* for a list of the Adaptive Server counters you can monitor.

sql server clock tick length

Summary Information	
Name in pre-11.0 release	**cclkrate**
Default value	Platform-specific
Range of values	Platform-specific minimum–1000000, in multiples of default value
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **sql server clock tick length** parameter specifies the duration of the server's clock tick, in microseconds. Both the default value and the minimum value are platform-specific. Adaptive Server rounds values up to an even multiple of *n*, where *n* is the platform-specific clock-tick default value. You can find the current values for **sql server clock tick length** by using **sp_helpconfig** or **sp_configure**.

In mixed-use applications with some CPU-bound tasks, decreasing the value of **sql server clock tick length** helps I/O-bound tasks. A value of 20,000 is reasonable for this. Shortening the clock tick length means that CPU-bound tasks will exceed its allotted time on the engine more frequently per unit of time, which allows other tasks greater access to the CPU. This may also marginally increase response times, because Adaptive Server runs its service tasks once per clock tick. Decreasing the clock tick length means that the service tasks will be run more frequently per unit of time.

Increasing **sql server clock tick length** favors CPU-bound tasks, because they execute longer between context switches. The maximum value of 1,000,000 may be appropriate for primarily CPU-bound applications. However, any I/O-bound tasks may suffer as a result. This can be mitigated somewhat by tuning **cpu grace time** (see "cpu grace time" on page 11-100) and **time slice** (see "time slice" on page 11-127).

➤ ***Note***

Changing the value of **sql server clock tick length** can have serious effects on Adaptive Server's performance. You should consult with Sybase Technical Support before resetting this value.

time slice

Summary Information	
Name in pre-11.0 release	**time slice**
Default value	100
Range of values	50–1000
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **time slice** parameter sets the number of clock ticks that Adaptive Server's scheduler allows a task to run for. If **time slice** is set too low, Adaptive Server may spend too much time switching between tasks, which increases response time. If it is set too high, CPU-intensive tasks might monopolize engines, which also increases response time.

See "cpu grace time" on page 11-100. Also, see Chapter 21, "How Adaptive Server Uses Engines and CPUs," and "Adaptive Server Execution Task Scheduling" on page 21-7 in the *Performance and Tuning Guide* for a more detailed discussion of task scheduling.

Use **sp_sysmon** to determine how the **time slice** parameter affects voluntary yields by Adaptive Server engines. See "Voluntary Yields" on page 24-24 in the *Performance and Tuning Guide*.

upgrade version

Summary Information	
Name in pre-11.0 release	**upgrade version**
Default value	1101
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The **upgrade version** parameter reports the version of the upgrade utility that upgraded your master device. The upgrade utility checks and modifies this parameter during an upgrade.

◆ ***WARNING!***

Although this parameter is configurable, you should not reset it. Doing so may cause serious problems with Adaptive Server.

You can determine whether an upgrade has been done on your master device by using the **upgrade version** parameter without specifying a value:

```
sp_configure "upgrade version"
```

Security Related

The parameters in this group configure security-related features.

allow procedure grouping

Summary Information	
Name in pre-11.0 release	N / A
Default value	1 (on)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **allow procedure grouping** parameter controls the ability to group stored procedures of the same name so that they can be dropped together with a single **drop procedure** statement. The default value of 1 allows stored procedure grouping. Set the option to 0 to prohibit stored procedure grouping. To run Adaptive Server in the **evaluated configuration**, you must prohibit stored procedure grouping by setting this option to 0. The evaluated configuration is the configuration of Adaptive Server that passed security evaluation at the C2 level. See **evaluated configuration** in the *Adaptive Server Glossary* for more information.

auditing

Summary Information	
Name in pre-11.0 release	N / A
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **auditing** configuration parameter enables or disables auditing for Adaptive Server. Set the **auditing** parameter to 1 to enable auditing; set the parameter to 0 to disable auditing.

audit queue size

Summary Information	
Name in pre-11.0 release	**audit queue size**
Default value	100
Range of values	1–65535
Status	Static
Display level	Intermediate
Required role	System Security Officer

The in-memory audit queue holds audit records generated by user processes until the records can be processed and written to the audit trail. A System Security Officer can change the size of the audit queue with the **audit queue size** configuration parameter. There is a trade-off between performance and risk that must be considered when you configure the queue size. If the queue is too large, records can remain in it for some time. As long as records are in the queue, they are at risk of being lost if the system crashes. However, if the queue is too small, it can become full repeatedly, which affects overall system performance: user processes that generate audit records sleep if the audit queue is full.

Following are some guidelines for determining how big your audit queue should be. You must also take into account the amount of auditing to be done at your site.

- The memory requirement for a single audit record is 424 bytes.
- The maximum number of audit records that can be lost in a system crash is the size of the audit queue (in records), plus 20. After records leave the audit queue they remain on a buffer page until they are written to the current audit table on the disk. The pages are flushed to disk every 20 records at the most (less if the audit process is not constantly busy).
- Although the memory requirement for a single audit record is 424 bytes, a record can be as small as 22 bytes when it is written to a data page.
- In the system audit tables, the *extrainfo* field and fields containing names are of variable length, so audit records that contain complete name information are generally larger.

The number of audit records that can fit on a page varies from 4 to as many as 80 or more. The memory requirement for the default audit queue size of 100 is approximately 42K.

current audit table

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	0–8
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **current audit table** parameter establishes the table where Adaptive Server writes audit rows. A System Security Officer can change the current audit table with **sp_configure**, using this syntax:

```
sp_configure "current audit table", n
  [, "with truncate"]
```

where *n* is an integer that determines the new current audit table, as follows:

- 1 means *sysaudits_01*, 2 means *sysaudits_02*, and so forth, up to 8, which means *sysaudits_08*.
- 0 is a special integer that tells Adaptive Server to set the current audit table to the next table. For example, if your installation has three audit tables, *sysaudits_01*, *sysaudits_02*, and *sysaudits_03*, Adaptive Server sets the current audit table to:
  - 2 if the current audit table is *sysaudits_01*
  - 3 if the current audit table is *sysaudits_02*
  - 1 if the current audit table is *sysaudits_03*

"with truncate" specifies that Adaptive Server should truncate the new table if it is not already empty. The **sp_configure** command fails if this option is not specified and the table is not empty.

➤ ***Note***

If Adaptive Server truncates the current audit table, and you have not archived the data, the table's audit records are lost. Be sure that the audit data is archived before using the **with truncate** option.

To execute **sp_configure** to change the current audit table, you must have the **sso_role** active. You can write a threshold procedure to change the current audit table automatically.

max roles enabled per user

Summary Information	
Name in pre-11.0 release	N / A
Default value	20
Range of values	10–127
Status	Static
Display level	Intermediate
Required role	System Security Officer

The **max roles enabled per user** parameter sets the number of roles you can activate per user session. The maximum number of roles you can activate per user session is 127.

The maximum number of roles server-wide is 1024. The maximum number of user-defined roles you can activate server-wide is 992. This is because the first 32 roles are reserved for Sybase system roles.

For information on user-defined roles, see Chapter 4, "Administering Roles," in the *Security Administration Guide.*

secure default login

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0 (followed by another parameter naming the default login)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **secure default login** parameter specifies a default login for all users who are pre-authenticated but who do not have a login in *master..syslogins*.

Establish the secure default login with this syntax:

```
sp_configure "secure default login", 0,
    default_login_name
```

where:

- **secure default login** is the name of the parameter.
- **0** is a required parameter because the 2nd parameter of **sp_configure** must be a numeric value.
- *default_login_name* is the name of the default login for a user who is unknown to Adaptive Server, but who has already been authenticated by a security mechanism. The login name must be a valid login in *master..syslogins*.

For example, to specify “dlogin” as the secure default login, type:

```
sp_configure "secure default login", 0, dlogin
```

select on syscomments.text column

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

This parameter enables protection of the text of database objects through restriction of the **select** permission on the *text* column of the *syscomments* table. The default value of 1 allows **select** permission to "public." Set the option to 0 to restrict **select** permission to the object owner and the System Administrator. In order to run Adaptive Server in the **evaluated configuration**, you must protect the text of database objects by setting this option to 0.

The evaluated configuration is the configuration of Adaptive Server that passed security evaluation at the C2 level. See **evaluated configuration** in the *Adaptive Server Glossary* for more information.

suspend audit when device full

Summary Information	
Name in pre-11.0 release	N / A
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **suspend audit when device full** parameter determines what Adaptive Server does when an audit device becomes completely full.

➤ ***Note***

If you have two or more audit tables, each on a separate device other than the master device, and you have a threshold procedure for each audit table segment, the audit devices should never become full. Only if a threshold procedure is not functioning properly would the "full" condition occur.

Choose one of these values:

- 0 – truncates the next audit table and starts using it as the current audit table when the current audit table becomes full. If you set the parameter to 0, you ensure that the audit process is never suspended. However, you incur the risk that older audit records will get lost if they have not been archived.
- 1 – suspends the audit process and all user processes that cause an auditable event. To resume normal operation, the System Security Officer must log in and set up an empty table as the current audit table. During this period, the System Security Officer is exempt from normal auditing. If the System Security Officer's actions would generate audit records under normal operation, Adaptive Server sends an error message and information about the event to the error log.

To run in the evaluated configuration, set this parameter to 1.

The evaluated configuration is the configuration of Adaptive Server that passed security evaluation at the C2 level. See **evaluated configuration** in the *Adaptive Server Glossary* for more information.

systemwide password expiration

Summary Information	
Name in pre-11.0 release	**password expiration interval**
Default value	0
Range of values	0–32767
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **systemwide password expiration** parameter sets the number of days that passwords remain in effect after they are changed. If **systemwide**

password expiration is set to 0, passwords do not expire. If it is set to a number greater than 0, all passwords expire after the specified number of days. An account's password is considered expired if an interval greater than *number_of_days* has passed since the last time the password for that account was changed.

When the number of days remaining before expiration is less than 25 percent of the value of **systemwide password expiration** or 7 days, whichever is greater, each time the user logs in, he or she is greeted with a message giving the number of days remaining before expiration. Users can change their passwords anytime before expiration.

When an account's password has expired, the user can still log into Adaptive Server but cannot execute any commands until he or she has used **sp_password** to change his or her password. If the user issues any command other than **sp_password**, he or she receives an error message, and the command fails. If the System Security Officer changes the user's password while the account is in **sp_password**-only mode, the account returns to normal once the new password is assigned.

This restriction applies only to login sessions established after the password has expired. Users who are logged in at the time their passwords expire are not affected until the next time they log in.

The default for **systemwide password expiration** is 0 (no expiration). This parameter can be set only by a System Security Officer.

unified login required (Windows NT Only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0, 1
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The **unified login required** parameter requires that all users who log into Adaptive Server be authenticated by Windows NT LAN Manager.

The **use security services** parameter must be 1 to use the unified login security service.

use security services (Windows NT Only)

Summary Information	
Name in pre-11.0 release	N / A
Default value	0
Range of values	0, 1
Status	Static
Display level	Intermediate
Required role	System Security Officer

The **use security services** parameter specifies that Adaptive Server will use security services provided by Windows NT LAN Manager. If the parameter is set to 0, unified login services with LAN Manager cannot be used. Because the parameter is static, you must restart Adaptive Server for it to take effect.

User Environment

The parameters in this group configure user environments.

number of user connections

Summary Information	
Name in pre-11.0 release	**user connections**
Default value	25
Range of values	5–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **number of user connections** parameter sets the maximum number of user connections that can be connected to Adaptive Server at the

same time. It does not refer to the maximum number of processes; that number depends not only on the value of this parameter but also on other system activity.

Upper Limit to the *maximum number of user connections*

The maximum allowable number of file descriptors per process is operating system-dependent; see the configuration documentation for your platform.

The number of file descriptors available for Adaptive Server connections is stored in the global variable *@@max_connections*. You can report the maximum number of file descriptors your system can use with this command:

```
select @@max_connections
```

The return value represents the maximum number of file descriptors allowed by the system for your processes, minus overhead. Overhead increases with the number of engines. For more information on how multiprocessing affects the number file descriptors available for Adaptive Server connections, see "Managing User Connections" on page 10-7.

In addition, you must reserve a number of connections for the following items, which you also set with configuration parameters:

- The database devices, including mirror devices
- Site handlers
- Network listeners

The following formula determines how high you can set the **number of user connections**, **number of devices**, **max online engines**, **number of remote sites**, and **max number network listeners** configuration parameters:

number of user connections + (**number of devices** * **max online engines** * 2) + **number of remote sites** + **max number network listeners** must not be greater than the value of *@@max_connections*.

Optimizing the Value of the *max number of user connections* Parameter

There is no formula for determining how many connections to allow for each user. You must estimate this number, based on the system and user requirements described here. You must also take into account that on a system with many users, there is more likelihood that connections needed only occasionally or transiently can be shared among users. The following processes require user connections:

- One connection is needed for each user running **isql**.
- Application developers use one connection for each editing session.
- The number of connections required by users running an application depends entirely on how the application has been programmed. Users executing Open Client programs need one connection for each open DB-Library **dbprocess** or Client-Library **cs_connection**.

➤ ***Note***

It is a good idea to estimate the maximum number of connections that will be used by Adaptive Server and to update **number of user connections** as you add physical devices or users to the system. Use **sp_who** periodically to determine the number of active user connections on your Adaptive Server.

Certain other configuration parameters, including **stack size** and **default network packet size**, affect the amount of memory for each user connection.

permission cache entries

Summary Information	
Name in pre-11.0 release	**cfgcprot**
Default value	15
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **permission cache entries** parameter determines the number of cache protectors per task. This parameter increases the amount of memory for each user connection and worker process.

Information about user permissions is held in the permission cache. When Adaptive Server needs to check permissions, it looks first in the permission cache: If it does not find what it needs, it looks in the *sysprotects* table. In terms of response time, it is significantly faster if

Adaptive Server finds the information it needs in the permission cache and does not have to read *sysprotects*.

However, Adaptive Server looks in the permission cache only when it is checking user permissions, not when permissions are being granted or revoked. When a permission is granted or revoked, the entire permission cache is flushed. This is because existing permissions have time stamps that become outdated when new permissions are granted or revoked.

If users on your Adaptive Server frequently perform operations that require their grantable or revokable permissions to be checked, you may see a small performance gain by increasing the value of **permission cache entries**. This effect is not likely to be significant enough to warrant extensive tuning.

If users on your Adaptive Server frequently grant or revoke permissions, avoid setting **permission cache entries** to a large value. The space used for the permission cache would be wasted, since the cache is flushed with each **grant** and **revoke** command.

stack guard size

Summary Information	
Name in pre-11.0 release	**cguardsz**
Default value	4096
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The **stack guard size** parameter sets the size (in bytes) of the stack guard area. The **stack guard area** is an overflow stack of configurable size at the end of each stack. Adaptive Server allocates one stack for each user connection and worker process when it starts. These stacks are located contiguously in the same area of memory, with a guard area at the end of each stack. At the end of each stack guard area is a **guardword**, which is a 4-byte structure with a known pattern. Figure 11-5 illustrates how a process can corrupt a stack guardword.

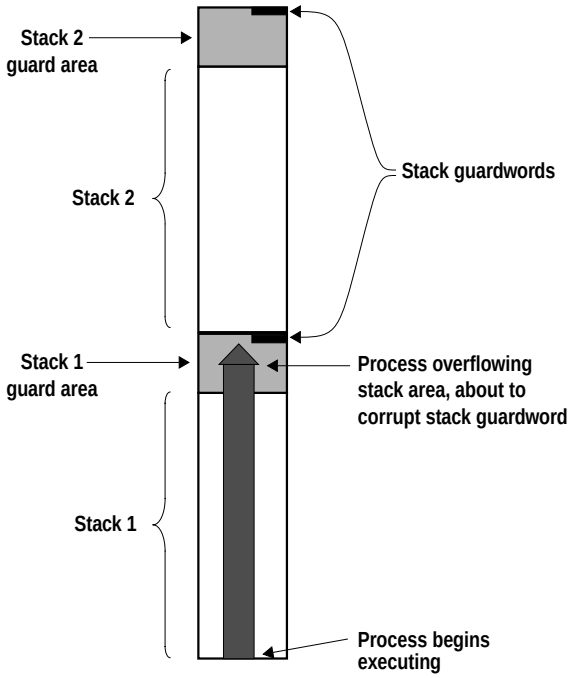


Figure 11-5: Process about to corrupt stack guardword

Adaptive Server periodically checks to see whether the stack pointer for a user connection has entered the stack guard area associated with that user connection's stack. If it has, Adaptive Server aborts the transaction, returns control to the application that generated the transaction, and generates Error 3626:

```
The transaction was aborted because it used too
much stack space.  Either use sp_configure to
increase the stack size, or break the query into
smaller pieces. spid: %d, suid: %d, hostname:
%.*s, application name: %.*s
```

Adaptive Server also periodically checks the guardword pattern to see if it has changed, thus indicating that a process has overflowed the stack boundary. When this occurs, Adaptive Server prints these messages to the error log and shuts down:

```
kernel: *** Stack overflow detected: limit: 0x%lx sp: 0x%lx
kernel: *** Stack Guardword corrupted
kernel: *** Stack corrupted, server aborting
```

In the first message, "limit" is the address of the end of the stack guard area, and "sp" is the current value of the stack pointer.

In addition, Adaptive Server periodically checks the stack pointer to see whether it is completely outside both the stack and the stack guard area for the pointer's process. If it is, Adaptive Server shuts down, even if the guardword is not corrupted. When this happens, Adaptive Server prints the following messages to the error log:

```
kernel: *** Stack overflow detected: limit: 0x%lx sp: 0x%lx
kernel: *** Stack corrupted, server aborting
```

The default value for **stack guard size** is appropriate for most applications. However, if you experience server shutdown from either stack guardword corruption or stack overflow, increase **stack guard size** by a 2K increment. **Each** configured user connection and worker process has a stack guard area; thus, when you increase **stack guard size**, you use up that amount of memory, multiplied by the number of user connections and worker processes you have configured.

Rather than increasing **stack guard size** to avoid stack overflow problems, consider increasing **stack size** (see "stack size" on page 11-143). The stack guard area is intended as an overflow area, not as an extension to the regular stack.

Adaptive Server allocates stack space for each task by adding the values of the **stack size** and **stack guard size** parameters. **stack guard size** must be configured in multiples of 2K. If the value you specify is not a multiple of 2K, **sp_configure** verification routines round the value up to the next highest multiple.

stack size

Summary Information	
Name in pre-11.0 release	**stack size**
Default value	platform-specific
Range of values	Platform-specific minimum–2147483647
Status	Static
Display level	Basic
Required role	System Administrator

The **stack size** parameter specifies the size (in bytes) of the execution stacks used by each user process on Adaptive Server. To find the **stack size** parameter values for your platform, use **sp_helpconfig** or **sp_configure**. **stack size** must be configured in multiples of 2K. If the value you specify is not a multiple of 2K, **sp_configure** verification routines round the value up to the next highest multiple.

An **execution stack** is an area of Adaptive Server memory in which user processes keep track of their process context and store local data.

Certain queries can contribute to the probability of a stack overflow. Examples include queries with extremely long **where** clauses, long select lists, deeply nested stored procedures, and multiple selects and updates using **holdlock**. When a stack overflow occurs, Adaptive Server prints an error message and rolls back the transaction. See “stack guard size” on page 11-140 for more information on stack overflows. See the *Adaptive Server Error Messages* manual for more information on specific error messages.

The two options for remedying stack overflows are to break the large queries into smaller queries and to increase **stack size**. Changing **stack size** affects the amount of memory required for **each** configured user connection and worker process. See “total memory” on page 11-92 for further information.

If you have queries that exceed the size of the execution stack, it may make sense to rewrite them as a series of smaller queries. This is particularly true if there are only a small number of such queries or if they are run infrequently.

There is no way to determine how much stack space a query will require without actually running the query. Stack space for each user connection and worker process is preallocated at start-up.

Therefore, determining the appropriate value for **stack size** is an empirical process. You should test your largest and most complex queries using the default value for **stack size**. If they run without generating error messages, the default is probably sufficient. If they generate error messages, you should begin by increasing **stack size** by a small amount (2K). Rerun your queries and see if the amount you have added is sufficient. If it is not, continue to increase **stack size** until queries run without generating error messages.

user log cache size

Summary Information	
Name in pre-11.0 release	N / A
Default value	2048
Range of values	2048–2147483647
Status	Static
Display level	Intermediate
Required role	System Administrator

The **user log cache size** parameter specifies the size (in bytes) for each user's log cache. There is one user log cache for each configured user connection and worker process. Adaptive Server uses these user log caches to buffer the user transaction log records, which reduces the contention at the end of the transaction log.

When a user log cache becomes full or another event occurs (such as when the transaction completes), Adaptive Server "flushes" all log records from the user log cache to the database transaction log. By first consolidating the log records in each user's log cache, rather than immediately adding each record to the database's transaction log, Adaptive Server reduces contention of processes writing to the log, especially for SMP systems configured with more than one engine.

➤ ***Note***

For transactions using a database with mixed data and log segments, the user log cache is flushed to the transaction log after each log record. No buffering takes place. If your databases do not have dedicated log segments, you should not increase the user log cache size.

You should not configure **user log cache size** to be larger than the maximum amount of log information written by an application's transaction. Since Adaptive Server flushes the user log cache when the transaction completes, any additional memory allocated to the user log cache is wasted. If no transaction in your server generates more than 4000 bytes of transaction log records, set **user log cache size** no higher than that value. For example:

```
sp_configure "user log cache size", 4000
```

Setting **user log cache size** too high wastes memory. Setting it too low can cause the user log cache to fill up and flush more than once per transaction, increasing the contention for the transaction log. If the volume of transactions is low, the amount of contention for the transaction log may not be significant.

Use **sp_sysmon** to understand how this parameter affects cache behavior. See "ULC Flushes to Transaction Log" on page 24-44 and "Maximum ULC Size" on page 24-46 in the *Performance and Tuning Guide*.

user log cache spinlock ratio

Summary Information	
Name in pre-11.0 release	N / A
Default value	20
Range of values	1–2147483647
Status	Static
Display level	Intermediate
Required role	System Administrator

For Adaptive Servers running with multiple engines, the **user log cache spinlock ratio** parameter specifies the ratio of user log caches per

user log cache **spinlock**. There is one user log cache for each configured user connection.

The default value for this parameter is 20, or one spinlock for each 20 user connections configured for your server.

Use **sp_sysmon** to understand how this parameter affects cache behavior. See "Spinlock Contention" on page 24-76 in the *Performance and Tuning Guide*.

For more information about configuring spinlock ratios, see "Configuring Spinlock Ratio Parameters" on page 10-9.

12 Limiting Access to Server Resources

This chapter describes how to use resource limits to restrict the I/O cost, row count, or processing time that an individual login or application can use during critical times. It also describes how to create named time ranges to specify contiguous blocks of time for resource limits. Topics include:

- What Are Resource Limits? 12-1
- Planning Resource Limits 12-2
- Enabling Resource Limits 12-2
- Defining Time Ranges 12-3
- Identifying Users and Limits 12-7
- Understanding Limit Types 12-13
- Creating a Resource Limit 12-17
- Getting Information on Existing Limits 12-19
- Modifying Resource Limits 12-21
- Dropping Resource Limits 12-22
- Resource Limit Precedence 12-24

What Are Resource Limits?

Adaptive Server provides resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. A **resource limit** is a set of parameters specified by a System Administrator to prevent an individual login or application from:

- Exceeding estimated or actual I/O costs, as determined by the optimizer
- Returning more than a set number of rows
- Exceeding a given elapsed time

The set of parameters for a resource limit includes the time of day to enforce the limit and the type of action to take. For example, you can prevent huge reports from running during critical times of the day or kill a session whose query produces unwanted **Cartesian products**.

Planning Resource Limits

In planning a resource limit, consider:

- When to impose the limit (times of day and days of the week)
- Which users and applications to monitor
- What type of limit to impose
  - I/O cost (either estimated or actual) for queries that may require large numbers of logical and physical reads
  - Row count for queries that may return large result sets
  - Elapsed time for queries that may take a long time to complete either because of their own complexity or because of external factors such as server load
- Whether to apply a limit to individual queries or to specify a broader scope (query batch or transaction)
- Whether to enforce the I/O cost limits prior to or during execution
- What action to take when the limit is exceeded (issue a warning, abort the query batch or transaction, or kill the session)

After completing the planning, use system procedures to perform the following actions:

- Specify times for imposing the limit by creating a named time range using **sp_add_time_range.**
- Create new resource limits using **sp_add_resource_limit.**
- Obtain information about existing resource limits using **sp_help_resource_limit.**
- Modify time ranges and resource limits using **sp_modify_time_range** and **sp_modify_resource_limit**, respectively.
- Drop time ranges and resource limits using **sp_drop_time_range** and **sp_drop_resource_limit**, respectively.

Enabling Resource Limits

To enforce resource limits, first configure Adaptive Server to enable them. Use the **allow resource limits** configuration parameter as follows:

```
sp_configure "allow resource limits", 1
```

1 enables the resource limits; 0 disables them. The **allow resource limits** configuration parameter is a static parameter, so you must restart the server to reset the changes.

The **allow resource limits** configuration parameter signals the server to allocate internal memory for time ranges, resource limits and internal server alarms, and internally assigns applicable ranges and limits to login sessions.

Setting **allow resource limits** to 1 also changes the output of **showplan** and **statistics I/O**, as follows:

- **showplan** displays estimated I/O cost information for DML statements. The information displayed is the optimizer's cost estimate for the query as a unitless number. The total estimated I/O cost is displayed for the query as a whole.
- **statistics io** includes the actual total I/O cost of a statement according to the optimizer's costing formula.

Defining Time Ranges

A **time range** is a contiguous block of time within a single day across one or more contiguous days of the week. It is defined by its starting and ending periods. Contiguous days of the week are allowed to wrap around the end of one week to the beginning of the next. In other words, Tuesday and Wednesday are contiguous days, just as are Sunday and Monday.

Adaptive Server includes one predefined time range, the "at all times" range, which covers the period midnight through midnight, Monday through Sunday. You can create, modify, and drop additional time ranges as necessary for resource limits.

Named time ranges may overlap. However, the limits for a particular user/application combination may not be associated with named time ranges that overlap. You can create different limits that share the same time range.

For example, assume that you limit "joe_user" to returning 100 rows when he is running the payroll application during business hours. Later, you attempt to limit his row retrieval during peak hours, which overlap with business hours. You will get a message that the new limit failed, because it would have overlapped with an existing limit.

Although you cannot limit the row retrieval for "joe_user" in the payroll application during overlapping time ranges, nothing stops

you from putting a second limit on "joe_user" during the same time range as the row retrieval limit. For example, you can limit the amount of time one of his queries can run to the same time range that you used to limit his row retrieval.

When you create a named time range, Adaptive Server stores it in the *systimeranges* system table to control when a resource limit is active. Each time range has a range ID number. The "at all times" range is range ID 1. Adaptive Server messages use the range ID number to refer to specific time ranges. For example, the message that the time range of a second resource limit would overlap with an existing limit includes the range ID number of the existing limit.

Determining What Time Ranges You Need

Use a chart like the one below to determine what time ranges to create for each server. Monitor server usage throughout the week; then indicate the periods when your server is especially busy or is performing crucial tasks that should not be interrupted.

Day / Time	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	00:00
Mon																									
Tues																									
Wed																									
Thurs																									
Fri																									
Sat																									
Sun																									

Creating Named Time Ranges

Create new time ranges with the **sp_add_time_range** system procedure. Use this system procedure's parameters to:

- Name the time range
- Specify the days of the week to begin and end the time range

- Specify the times of the day to begin and end the time range

For syntax and detailed information about this system procedure, see **sp_add_time_range** in the *Adaptive Server Reference Manual*.

A Time Range Example

Assume that two critical jobs are scheduled to run every week at the following times:

- Job 1 runs from 07:00 to 10:00 on Tuesday and Wednesday.
- Job 2 runs from 08:00 on Saturday to 13:00 on Sunday.

The following table uses "1" to indicate when job 1 runs and "2" to indicate when job 2 runs:.

Day / Time	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	00:00
Mon																									
Tues								1	1	1	1														
Wed								1	1	1	1														
Thurs																									
Fri																									
Sat									2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Sun	2	2	2	2	2	2	2	2	2	2	2	2	2	2											

Job 1 can be covered by a single time range, *tu_wed_7_10*:

```
sp_add_time_range tu_wed_7_10, tuesday, wednesday,
"7:00", "10:00"
```

Job 2, however, requires two separate time ranges, for Saturday and Sunday:

```
sp_add_time_range saturday_night, saturday,
saturday, "08:00", "23:59"

sp_add_time_range sunday_morning, sunday, sunday,
"00:00", "13:00"
```

Modifying a Named Time Range

Change a time range's starting and ending date and time with the **sp_modify_time_range** system procedure. Use this system procedure's parameters to:

- Specify which time range to modify
- Specify the change to the days of the week
- Specify the change to the times of the day

For syntax and detailed information about this system procedure, see **sp_modify_time_range** in the *Adaptive Server Reference Manual*.

For example, to change the end day of the *business_hours* time range to Saturday, retaining the existing start day, start time, and end time., issue the following command:

```
sp_modify_time_range business_hours, NULL,
Saturday, NULL, NULL
```

To specify a new end day and end time for the *before_hours* time range, issue the following command:

```
sp_modify_time_range before_hours, NULL, Saturday,
NULL, "08:00"
```

➤ ***Note***

You cannot modify the "at all times" time range.

Dropping a Named Time Range

Drop a user-defined time range with the **sp_drop_time_range** system procedure. Use this system procedure's parameter to specify which time range to drop.

For syntax and detailed information about this system procedure, see **sp_drop_time_range** in the *Adaptive Server Reference Manual*.

For example, to remove the *evenings* time range from the *systimeranges* system table in the *master* database, issue the following command:

```
sp_drop_time_range evenings
```

➤ ***Note***

You cannot drop the “at all times” time range or any time range for which resource limits are defined.

When Do Time Range Changes Take Effect?

The active time ranges are bound to a login session at the beginning of each query batch. A change in the server’s active time ranges due to a change in actual time has no effect on a session during the processing of a query batch. In other words, if a resource limit restricts query batches during a given time range, but the query batch begins before that time range becomes active, the query batch that is already running is not affected by the resource limit. However, if you run a second query batch during the same login session, that query batch will be affected by the change in time.

Adding, modifying, and deleting time ranges through system procedures does not affect the active time ranges for the login sessions currently in progress.

Modifying a resource limit that has a transaction as its scope does not affect any transactions currently in progress.

If a resource limit has a transaction as its scope, and a change occurs in the server’s active time ranges while a transaction is running, the newly active time range does not affect the transaction currently in progress.

Identifying Users and Limits

For each resource limit, you must specify the object to which the limit applies.

You can apply a resource limit to any of the following:

- All applications used by a particular login
- All logins that use a particular application
- A specific application used by a particular login

where **application** is defined as a client program running on top of Adaptive Server, accessed through a particular login. To run an application on Adaptive Server, you must specify its name through the CS_APPNAME connection property using **cs_config** (an Open Client Client-Library application) or the DBSETLAPP function in

Open Client DB-Library. To list named applications running on your server, select the *program_name* column from the *master..sysprocesses* table.

For more information about the CS_APPNAME connection property, see the *Open Client Client-Library/C Reference Manual*. For more information on the DBSETLAPP function, see the *Open Client DB-Library/C Reference Manual*.

Identifying Heavy-Usage Users

Before you implement resource limits, run the **sp_reportstats** system procedure. The output from this procedure will help you identify the users with heavy system usage. For example:

```
sp_reportstats
```

```
Name    Since          CPU     Percent CPU  I/O     Percent I/O
------  -----------    -----   ------------ -----   -------------
probe   jun 19 1993    0       0%           0       0%
julie   jun 19 1993    10000   24.9962%     5000    24.325%
jason   jun 19 1993    10002   25.0013%     5321    25.8866%
ken     jun 19 1993    10001   24.9987%     5123    24.9234%
kathy   jun 19 1993    10003   25.0038%     5111    24.865%

      Total CPU   Total I/O
      ---------   ---------
      40006       20555
```

The output above indicates that usage is balanced among the users. For more information on chargeback accounting, see “cpu accounting flush interval” on page 11-99 and “i/o accounting flush interval” on page 11-108.

Identifying Heavy-Usage Applications

To identify the applications running on your system and the users who are running them, query the *sysprocesses* system table in the *master* database.

The following query determines that **isql**, **payroll**, **perl**, and **acctng** are the only client programs whose names were passed to the Adaptive Server:

```
select spid, cpu, physical_io,
  substring(user_name(uid),1,10) user_name,
  hostname, program_name, cmd
from sysprocesses
```

```
spid  cpu    physical_io  user_name hostname program_name cmd
----  ---    -----------  --------- -------- ------------ ------
  17    4          12748  dbo       sabrina  isql         SELECT
 424    5              0  dbo       HOWELL   isql         UPDATE
 526    0            365  joe       scotty   payroll      UPDATE
 568    1           8160  dbo       smokey   perl         SELECT
 595   10              1  dbo       froth    isql         DELETE
 646    1              0  guest     walker   isql         SELECT
 775    4          48723  joe_user  mohindra acctng       SELECT

(7 rows affected)
```

Because *sysprocesses* is built dynamically to report current processes, repeated queries produce different results. Repeat this query throughout the day over a period of time to determine which applications are running on your system.

The CPU and physical I/O values are flushed to the *syslogins* system table periodically where they increment the values shown by **sp_reportstats**.

After identifying the applications running on your system, use **showplan** and **statistics io** to evaluate the resource usage of the queries in the applications.

If you have configured Adaptive Server to enable resource limits, you can use **showplan** to evaluate resources used prior to execution and **statistics io** to evaluate resources used during execution. For information on configuring Adaptive Server to enable resource limits, see "Enabling Resource Limits" on page 12-2.

showplan prints the total estimated I/O cost for each query in a batch. The information displayed is the optimizer's cost estimate for the query, based on logical and physical accesses. This cost estimate is dependent on the table statistics (number and distribution of values) and the size of the appropriate buffer pools. It is independent of such factors as the state of the buffer pools and the number of active users. For more information, see "showplan Messages Describing Access Methods, Caching, and I/O Cost" in the *Performance and Tuning Guide*.

statistics io displays the total actual I/O for each query. This value is a number representing the sum of the number of logical I/Os multiplied by the cost of a logical I/O and the number of physical I/Os multiplied by the cost of a physical I/O. For more information on these numbers, see "How Is "Fast" Determined?" in the *Performance and Tuning Guide*.

In addition to **statistics io**, **statistics time** is also useful for evaluating the resources a query consumes. Use **statistics time** to display the time it

takes to execute each step of the query. For more information, see "Diagnostic Tools for Query Optimization" on page 8-6 in the *Performance and Tuning Guide*.

Choosing a Limit Type

After you determine the users and applications to limit, you have a choice of three different types of resource limits. Each is designed to limit resource usage in a different way.

The following table describes the function and scope of each limit type and indicates the tools that help determine whether a particular query might benefit from this type of limit. In some cases, it may be appropriate to create more than one type of limit for a given user and application. For more information on limit types, see "Understanding Limit Types" on page 12-13.

Table 12-1: Resource limit types

Limit Type	Use for Queries That	Measuring Resource Usage	Scope	Enforced During
io_cost	Require many logical and physical reads	Use **set showplan on** before running the query, to display its estimated I/O cost; use **set statistics io on** to observe the actual I/O cost.	Query	Pre-execution or execution
row_count	Return large result sets	Use the *@@rowcount* global variable to help develop appropriate limits for row count.	Query	Execution
elapsed_time	Take a long time to complete, either because of their own complexity or because of external factors such as server load or waiting for a lock	Use **set statistics time on** before running the query, to display elapsed time in milliseconds.	Query batch or transaction	Execution

The *spt_limit_types* system table stores information about each limit type.

Determining Time of Enforcement

Time of enforcement is the phase of query processing during which Adaptive Server applies a given resource limit. Resource limits occur during:

- Pre-execution time – Adaptive Server applies resource limits prior to execution, based on the optimizer's I/O cost estimate. This limit prevents execution of potentially expensive queries. I/O cost is the only resource type that can be limited at pre-execution time.

  When evaluating the I/O cost of data manipulation language (DML) statements within the clauses of a conditional statement, Adaptive Server considers each DML statement individually. It evaluates all statements, even though only one clause will actually be executed.

  A pre-execution time resource limit can have only a query limit scope; that is, the values of the resources being limited at compile time are computed and monitored on a query-by-query basis only.

  Adaptive Server does not enforce pre-execution time resource limits statements in a trigger.

- Execution time – Adaptive Server applies resource limits at run time, and is usually used to prevent a query from monopolizing the server's (and operating system's) resources. Execution time limits may use more resources (additional CPU time as well as I/O) than pre-execution time limits.

Determining the Scope of Resource Limits

The *scope* parameter specifies the duration of a limit in Transact-SQL statements. The possible limit scopes are query, query batch, and transaction:

- Query – Adaptive Server applies resource limits to any single Transact-SQL statement that accesses the server; for example, **select**, **insert**, and **update**. When you issue these statements within a query batch, Adaptive Server evaluates them individually.

  Adaptive Server considers a stored procedure to be a series of DML statements. It evaluates the resource limit of each statement within the stored procedure. If a stored procedure executes another stored procedure, Adaptive Server evaluates

each DML statement within the nested stored procedure at the inner nesting level.

Adaptive Server checks pre-execution time resource limits with a query scope, one nesting level at a time. As Adaptive Server enters each nesting level, it checks the active resource limits against the estimated resource usage of each DML statement prior to executing any of the statements at that nesting level. A resource limit violation occurs if the estimated resource usage of any DML query at that nesting level exceeds the limit value of an active resource limit. Adaptive Server takes the action that is bound to the violated resource limit.

Adaptive Server checks execution time resource limits with a query scope against the cumulative resource usage of each DML query. A limit violation occurs when the resource usage of a query exceeds the limit value of an active execution time resource limit. Again, Adaptive Server takes the action that is bound to that resource limit.

- Query batch – A query batch consists of one or more Transact-SQL statements; for example, in **isql**, a group of queries becomes a query batch when executed by a single **go** command terminator.

  The query batch begins at nesting level 0; each call to a stored procedure increments the nesting level by 1 (up to the maximum nesting level). Each return from a stored procedure decrements the nesting level by 1.

  Only execution time resource limits can have a query batch scope.

  Adaptive Server checks execution time resource limits with a query batch scope against the cumulative resource usage of the statements in each query batch. A limit violation occurs when the resource usage of the query batch exceeds the limit value of an active execution time resource limit. Adaptive Server takes the action that is bound to that resource limit.

- Transaction – Adaptive Server applies limits with a transaction scope to all nesting levels during the transaction against the cumulative resource usage for the transaction.

  A limit violation occurs when the resource usage of the transaction exceeds the limit value of an active execution time resource limit. Adaptive Server takes the action that is bound to that resource limit.

Only execution time resource limits can have a transaction scope.

Adaptive Server does not recognize nested transactions when applying resource limits. A resource limit on a transaction begins when *@@trancount* is set to 1 and ends when *@@trancount* is set to 0.

Understanding Limit Types

There are three types of resource limits that allow you to limit resource usage in different ways.

Limiting I/O Cost

I/O cost is based on the number of logical and physical accesses ("reads") used during query processing. To determine the most efficient processing plan prior to execution, the Adaptive Server optimizer uses both logical and physical resources to compute an estimated I/O cost.

Adaptive Server uses the result of the optimizer's costing formula as a "unitless" number; that is, a value not necessarily based on a single unit of measurement (such as seconds or milliseconds).

In order to set resource limits, you must understand how those limits translate into run-time system overhead. For example, you must know the effect that a query with a cost of *x* logical and of *y* physical I/Os has on a production server.

Limiting **io_cost** can control I/O intensive queries, including queries that return a large result set. However, if you run a simple query that returns all the rows of a large table, and you do not have current statistics on the table's size, the optimizer may not estimate that the query will exceed the **io_cost** resource limit. To prevent queries from returning large result sets, create a resource limit on **row_count**.

The tracking of I/O cost limits may be less precise for partitioned tables than for unpartitioned tables when Adaptive Server is configured for parallel query processing. For more information on using resource limits in parallel queries, see Chapter 14, "Parallel Query Optimization," in the *Performance and Tuning Guide*.

Identifying I/O Costs

To develop appropriate limits for I/O cost, determine the number of logical and physical reads required for some typical queries. Use the following **set** commands:

- **set showplan on** displays the optimizer's cost estimate. Use this information to set pre-execution time resource limits. A pre-execution time resource limit violation occurs when the optimizer's I/O cost estimate for a query exceeds the limit value. Such limits prevent the execution of potentially expensive queries.
- **set statistics io on** displays the number of actual logical and physical reads required. Use this information to set execution time resource limits. An execution time resource limit violation occurs when the actual I/O cost for a query exceeds the limit value.

Statistics for actual I/O cost include access costs only for user tables and worktables involved in the query. Adaptive Server may use other tables internally; for example, it accesses *sysmessages* to print out statistics. Therefore, there may be instances when a query exceeds its actual I/O cost limit, even though the statistics indicate otherwise.

In costing a query, the optimizer assumes that every page it thinks will be needed will require a physical I/O for the first access and will be found in the cache for repeated accesses. Actual I/O costs may differ from the optimizer's estimated costs, for several reasons.

The estimated cost will be higher than the actual cost if some pages are already in the cache or if the statistics are incorrect. The estimated cost may be lower than the actual cost if the optimizer chooses 16K I/O, and some of the pages are in 2K cache pools, which requires many 2K I/Os. Also, if a big join forces the cache to flush its pages back to disk, repeated access may require repeated physical I/Os.

You cannot expect the optimizer's estimates to be accurate if the distribution or density statistics are out of date or cannot be used.

Calculating the I/O Cost of a Cursor

The cost estimate for processing a cursor is calculated at **declare cursor** time for all cursors except execute cursors (cursors declared for execution of a stored procedure that contains a single **select**). The cost estimate for an execute cursor is calculated when the cursor opens.

Pre-execution time resource limits on I/O cost are enforced at **open** *cursorname* time for all cursor types. The optimizer recalculates the limit value each time the user attempts to open the cursor.

An execution time resource limit applies to the cumulative I/O cost of a cursor from the time the cursor opens to the time it closes. The optimizer recalculates the I/O limit each time a cursor opens.

For a discussion of cursors, see Chapter 17, "Cursors: Accessing Data Row by Row," in the *Transact-SQL User's Guide*.

The Scope of the *io_cost* Limit Type

A resource limit that restricts I/O cost applies only to single queries. If you issue several statements in a query batch, Adaptive Server evaluates the I/O usage for each query. For more information, see "Determining the Scope of Resource Limits" on page 12-11.

Limiting Elapsed Time

Elapsed time is the number of seconds, in wall-clock time, required to execute a query batch or transaction. Elapsed time is determined by such factors as query complexity, server load, and waiting for locks.

Use information gathered with **set statistics time** to help develop appropriate limits for elapsed time. You can limit the elapsed time resource only at execution time.

With **set statistics time** set **on**, run some typical queries to determine processing time in milliseconds. Remember to convert the milliseconds to seconds when you create the resource limit.

Elapsed time resource limits are applied to all SQL statements in the limit's scope (query batch or transaction), not just to the DML statements. A resource limit violation occurs when the elapsed time for the appropriate scope exceeds the limit value.

Because elapsed time is limited only at execution time, an individual query will continue to run, even if its elapsed time exceeds the limit. If there are multiple statements in a batch, an elapsed time limit takes effect after a statement violates the limit and before the next statement is executed. If there is only one statement in a batch, setting an elapsed time limit has no effect.

Separate elapsed time limits are not applied to nested stored procedures or transactions. In other words, if one transaction is

nested within another, the elapsed time limit applies to the outer transaction, which encompasses the elapsed time of the inner transaction. Therefore, if you are counting the wall-clock running time of a transaction, that running time includes all nested transactions.

The Scope of the *elapsed_time* Limit Type

The scope of a resource limit that restricts elapsed time is either a query batch or transaction. You cannot restrict the elapsed time of a single query. For more information, see "Determining the Scope of Resource Limits" on page 12-11.

Limiting the Size of the Result Set

The **row_count** limit type limits the number of rows returned to the user. A limit violation occurs when the number of rows returned by a **select** statement exceeds the limit value.

If the resource limit issues a warning as its action, and a query exceeds the row limit, the full number of rows are returned, followed by a warning that indicates the limit value; for example:

```
Row count exceeded limit of 50.
```

If the resource limit's action aborts the query batch or transaction or kills the session, and a query exceeds the row limit, only the limited number of rows are returned and the query batch, transaction, or session aborts. Adaptive Server displays a message like the following:

```
Row count exceeded limit of 50.
Transaction has been aborted.
```

The **row_count** limit type applies to all **select** statements at execution time. You cannot limit an estimated number of rows returned at pre-execution time.

Determining Row Count Limits

Use the *@@rowcount* global variable to help develop appropriate limits for row count. Selecting this variable after running a typical query can tell you how many rows the query returned.

Applying Row Count Limits to a Cursor

A row count limit applies to the cumulative number of rows that are returned through a cursor from the time the cursor opens to the time it closes. The optimizer recalculates the **row_count** limit each time a cursor opens.

The Scope of the *row_count* Limit Type

A resource limit that restricts row count applies only to single queries, not to cumulative rows returned by a query batch or transaction. For more information, see "Determining the Scope of Resource Limits" on page 12-11.

Creating a Resource Limit

Create a new resource limit with the **sp_add_resource_limit** system procedure. The syntax of the **sp_add_resource_limit** system procedure is:

```
sp_add_resource_limit name, appname, rangename,
    limittype, limit_value, enforced, action, scope
```

Use this system procedure's parameters to:

- Specify the name of the user or application to which the resource limit applies.

  You must specify either a *name* or an *appname* or both. If you specify a user, the name must exist in the *syslogins* table. Specify "null" to create a limit that applies to all users or all applications.

- Specify the time range.

  The time range must already exist when you create the limit. For more information, see "Defining Time Ranges" on page 12-3.

- Specify the type of limit (**io_cost**, **row_count**, or **elapsed_time**), and set an appropriate value for the limit type.

  For more information, see "Choosing a Limit Type" on page 12-10.

- Specify whether the resource limit is enforced prior to or during query execution.

  Specify values for this parameter numerically. Pre-execution time resource limits, which are specified as 1, are valid only for the **io_cost** limit. Execution time resource limits, which are specified as 2, are valid for all three limit types. For more

information, see "Determining Time of Enforcement" on page 12-11.

- Specify the action to be taken (issue a warning, abort the query batch, abort the transaction, or kill the session).

  Specify numeric values for this parameter.

- Specify the scope (query, query batch, or transaction).

  Specify numeric values for this parameter. For more information, see "Determining the Scope of Resource Limits" on page 12-11.

For detailed information about this system procedure, see **sp_add_resource_limit** in the *Adaptive Server Reference Manual*.

Resource Limit Examples

This section includes three examples of setting resource limits.

Example 1

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 120, 2, 1, 2
```

This example creates a resource limit that applies to all users of the **payroll** application because the name parameter is NULL. The limit is valid during the *tu_wed_7_10* time range. The limit type, **elapsed_time**, is set to a value of 120 seconds. Because **elapsed_time** is enforced only at execution time, the *enforced* parameter is set to 2. The *action* parameter is set to 1, which issues a warning. The limit's *scope* is set to 2, query batch, by the last parameter. Therefore, when the elapsed time of the query batch takes more than 120 seconds to execute, Adaptive Server issues a warning.

Example 2

```
sp_add_resource_limit joe_user, NULL,
saturday_night, row_count, 5000, 2, 3, 1
```

This example creates a resource limit that applies to all ad hoc queries and applications run by "joe_user" during the *saturday_night* time range. If a query (*scope* = 1) returns more than 5000 rows, Adaptive Server aborts the transaction (*action* = 3). This resource limit is enforced at execution time (*enforced* = 2).

Example 3

```
sp_add_resource_limit joe_user, NULL, "at all
times", io_cost, 650, 1, 3, 1
```

This example also creates a resource limit that applies to all ad hoc queries and applications run by "joe_user". However, this resource limit specifies the default time range, "at all times". When the optimizer estimates that the **io_cost** of the query (*scope* = 1) would exceed the specified value of 650, Adaptive Server aborts the transaction (*action* = 3). This resource limit is enforced at pre-execution time (*enforced* = 1).

Getting Information on Existing Limits

Use the **sp_help_resource_limit** system procedure to get information about existing resource limits.

Users who do not have the System Administrator role can use **sp_help_resource_limit** to list their own resource limits (only).

Users either specify their own login names as a parameter or specify the *name* parameter as "null". Both of the following procedures return all resource limits for user "joe_user" when executed by joe_user:

```
sp_help_resource_limit
```

or

```
sp_help_resource_limit joe_user
```

System Administrators can use **sp_help_resource_limit** to get the following information:

- All limits as stored in *sysresourcelimits* (all parameters NULL); for example:

  ```
  sp_help_resource_limit
  ```

- All limits for a given login (*name* is not NULL, all other parameters are NULL); for example:

  ```
  sp_help_resource_limit joe_user
  ```

- All limits for a given application (*appname* is not NULL; all other parameters are NULL); for example:

  ```
  sp_help_resource_limit NULL, payroll
  ```

- All limits in effect at a given time or day (either *limittime* or *limitday* is not NULL; all other parameters NULL); for example:

```
sp_help_resource_limit @limitday = wednesday
```

- Limit, if any, in effect at a given time for a given login (*name* is not NULL, either *limittime* or *limitday* is not NULL); for example:

  ```
  sp_help_resource_limit joe_user, NULL, NULL,
  wednesday
  ```

For detailed information about this system procedure, see **sp_help_resource_limit** in the *Adaptive Server Reference Manual*.

Example of Listing All Existing Resource Limits

When you use **sp_help_resource_limit** without any parameters, Adaptive Server lists all resource limits within the server. For example:

```
sp_help_resource_limit
```

```
name      appname rangename rangeid limitid limitvalue enforced  action scope
----      ------- --------- ------- ------- ---------- --------  ------ -----
NULL      acctng  evenings        4       2        120         2       1     2
stein     NULL    weekends        1       3       5000         2       1     1
joe_user  acctng  bus_hours       5       3       2500         2       2     1
joe_user  finance bus_hours       5       2        160         2       1     6
wong      NULL    mornings        2       3       2000         2       1     1
wong      acctng  bus_hours       5       1         75         1       3     1
```

In the output, the *rangeid* column prints the value from *systimeranges.id* that corresponds to the name in the *rangename* column. The *limitvalue* column reports the value set by **sp_add_resource_limit** or **sp_modify_resource_limit**. Table 12-2 shows the meaning of the values in the *limitid*, *enforced*, *action*, and *scope* columns.

Table 12-2: Values for *sp_help_resource_limit* output

Column	Meaning	Value
limitid	What kind of limit is it?	1 I/O cost 2 Elapsed time 3 Row count
enforced	When is the limit enforced?	1 Prior to execution 2 During execution 3 Both
action	What action is taken when the limit is hit?	1 Issue a warning 2 Abort the query batch 3 Abort the transaction 4 Kill the session

Table 12-2: Values for *sp_help_resource_limit* output

Column	Meaning	Value
scope	What is the scope of the limit?	1 Query 2 Query batch 4 Transaction 6 Query batch + transaction

If a System Administrator specifies a login name when executing **sp_help_resource_limit**, Adaptive Server lists all resource limits for that login. The output displays not only resource limits specific to the named user, but all resource limits that pertain to all users of specified applications, because the named user is included among all users.

For example, the following output shows all resource limits that apply to a user named "joe_user". Because a resource limit is defined for all users of the **acctng** application, this limit is included in the output.

```
sp_help_resource_limit joe_user

name     appname rangename rangeid limitid limitvalue enforced  action scope
----     ------- --------- ------- ------- ---------- --------  ------ -----
NULL     acctng  evenings        4       2        120         2       1     2
joe_user acctng  bus_hours       5       3       2500         2       2     1
joe_user finance bus_hours       5       2        160         2       1     6
```

Modifying Resource Limits

Use the system procedure **sp_modify_resource_limit** to specify a new limit value or a new action to take when the limit is exceeded or both. You cannot change the login or application to which a limit applies or specify a new time range, limit type, enforcement time, or scope.

The syntax of **sp_modify_resource_limit** is:

```
sp_modify_resource_limit name, appname, rangename,
    limittype, limitvalue, enforced, action, scope
```

To modify a resource limit, specify the following values:

- You must specify non-null values for *rangename* and *limittype*. If necessary to uniquely identify the limit, specify non-null values for *action* and *scope*.
- You must specify a non-null value for either *name* or *appname*.

  - To modify a limit that applies to all users of a particular application, specify a *name* of "null".
  - To modify a limit that applies to all applications used by *name*, specify an *appnam*e of "null".
  - To modify a limit that governs a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

- Specifying "null" for *limitvalue* or *action* indicates that its value does not change.

For detailed information about this system procedure, see **sp_modify_resource_limit** in the *Adaptive Server Reference Manual*.

Examples of Modifying a Resource Limit

```
sp_modify_resource_limit NULL, payroll,
tu_wed_7_10, elapsed_time, 90, null, null, 2
```

This example changes the value of the resource limit that restricts elapsed time to all users of the *payroll* application during the *tu_wed_7_10* time range. The limit value for elapsed time decreases to 90 seconds (from 120 seconds). The values for time of execution, action taken, and scope remain unchanged.

```
sp_modify_resource_limit joe_user, NULL,
saturday_night, row_count, NULL, NULL, 2, NULL
```

This example changes the action taken by the resource limit that restricts the row count of all ad hoc queries and applications run by "joe_user" during the *saturday_night* time range. The previous value for action was 3, which aborts the transaction when a query exceeds the specified row count. The new value is being set to 2, which aborts the query batch. The values for limit type, time of execution, and scope remain unchanged.

Dropping Resource Limits

Use the system procedure **sp_drop_resource_limit** to drop a resource limit from an Adaptive Server.

The syntax for **sp_drop_resource_limit** is:

```
sp_drop_resource_limit {name , appname } [,
   rangename, limittype, enforced, action, scope]
```

To drop a resource limit, specify enough information to uniquely identify the limit. You must specify a non-null value for either *name* or *appname*. In addition, specify values according to those shown in Table 12-3.

Table 12-3: Identifying resource limits to drop

Parameter	Value Specified	Consequence
name	• Specified login	Drops limits that apply to the particular login.
	• NULL	Drops limits that apply to all users of a particular application.
appname	• Specified application	Drops limits that apply to a particular application.
	• NULL	Drops limits that apply to all applications used by the specified login.
timerange	• An existing time range stored in the *systimeranges* system table	Drops limits that apply to a particular time range.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *limittype*, enforcement time, *action*, and *scop*e, without regard to *rangename*.
limittype	• One of the three limit types: **row_count**, **elapsed_time**, **io_cost**	Drops limits that apply to a particular limit type.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *action*, and *scop*e, without regard to *limittype*.
enforced	• One of the enforcement times: pre-execution or execution	Drops the limits that apply to the specified enforcement time.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *limittype*, *timerange*, *action*, and *scop*e, without regard to enforcement time.
action	• One of the four action types: issue warning, abort query batch, abort transaction, kill session	Drops the limits that apply to a particular action type.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *limittype*, enforcement time, and *scop*e, without regard to *action*.

Table 12-3: Identifying resource limits to drop (continued)

Parameter	Value Specified	Consequence
scope	• One of the scope types: query, query batch, transaction • NULL	Drops the limits that apply to a particular scope. Drops all resource limits for the specified *name*, *appname*, *timerange*, *limittype*, enforcement time, and *action*, without regard to *scope*.

For detailed information about this system procedure, see **sp_drop_resource_limit** in the *Adaptive Server Reference Manual*.

Examples of Dropping a Resource Limit

```
sp_drop_resource_limit NULL, payroll, tu_wed_7_10
```

This example drops all resource limits for all users of the payroll application during the *tu_wed_7_10* time range.

```
sp_drop_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time
```

This example is similar to the preceding example, but drops only the resource limit that governs elapsed time for all users of the payroll application during the *tu_wed_7_10* time range.

```
sp_drop_resource_limit joe_user
```

This example drops all resource limits for the user named “joe_user”.

Consequences of Dropping a Login

When you use **sp_droplogin** to drop an Adaptive Server login, all resource limits associated with that login are also dropped.

Resource Limit Precedence

Adaptive Server provides precedence rules for time ranges and resource limits.

Time Ranges

For each login session during the currently active time ranges, only one limit can be active for each distinct combination of limit type, enforcement time, and scope. The precedence rules for determining the active limit are as follows:

- If no limit is defined for the login ID for either the "at all times" range or the currently active time ranges, there is no active limit.
- If limits are defined for the login for both the "at all times" and time-specific ranges, then the limit for the time-specific range takes precedence. Otherwise, the limit that is defined (either for the "at all times" or currently active time ranges) takes precedence.

Resource Limits

Since either the user's login name or the application name, or both, are used to identify a resource limit, Adaptive Server observes a predefined search precedence while scanning the *sysresourcelimits* table for applicable limits for a login session. The following table describes the precedence of matching ordered pairs of login name and application name:

Level	Login Name	Application Name
1	joe_user	**payroll**
2	NULL	**payroll**
3	joe_user	NULL

If one or more matches are found for a given precedence level, no further levels are searched. This prevents conflicts regarding similar limits for different login / application combinations.

If no match is found at any level, no limit is imposed on the session.

13 Configuring Character Sets, Sort Orders, and Languages

This chapter discusses Adaptive Server internationalization and localization support issues. Topics include:

- Language Support for International Installations 13-1
- Character Sets and Sort Orders 13-2
- Software Messages 13-4
- Changing the Default Character Set, Sort Order, or Language 13-6
- Installing Date Strings for Unsupported Languages 13-15

Language Support for International Installations

Sybase provides both internationalization and localization support for international installations. **Internationalization** is the process of designing software products so that a single version can be adapted to different languages or regions, conforming to local requirements and customs without engineering changes. Adaptive Server is internationalized so that it can correctly process the characters used in different languages. The Sybase Character Sets product provides the character set definition files and sort order definition files required for data processing support for the major business languages in Western Europe, Eastern Europe, the Middle East, Latin America, and Asia. By default, Adaptive Server comes with the supported character sets and sort orders for the Western European languages.

Localization is the adaptation of an internationalized product to meet the requirements of one particular language or region, including providing translated system messages and correct formats for date, time, and currency. The Sybase Language Modules provide translated system messages and formats for the following languages: Chinese (Simplified), French, German, Japanese, and Spanish. By default, Adaptive Server comes with U.S. English message files.

This chapter describes the character sets and language modules and summarizes the steps needed to change the default character set, sort order, or message language for Adaptive Server.

Character Sets and Sort Orders

If you require support for processing data in a language other than a Western European language, you need to purchase and install the Sybase Character Sets product.

Types of Internationalization Files

The files that support data processing in a particular language are called **internationalization files**. Several types of internationalization files come with Adaptive Server and the Sybase Character Sets product. These files are described in Table 13-1.

Table 13-1: Internationalization files

File	Location	Purpose and Contents
charset.loc	In each character set subdirectory of the *charsets* directory	Character set definition files that define the lexical properties of each character, such as alphanumeric, punctuation, operand, and uppercase or lowercase. Used by the Adaptive Server to correctly process data.
**.srt*	In each character set subdirectory of the *charsets* directory	Defines the sort order for alphanumeric and special characters, including ligatures, diacritics, and other language-specific considerations.
**.xlt*	In each character set subdirectory of the *charsets* directory	Terminal-specific character translation files for use with utilities such as **bcp** and **isql**. These files are part of the language module for Open Client. For more information about how the *.xlt* files are used, see Chapter 14, "Configuring Client/Server Character Set Conversions," and the the *Utility Programs* manual for your platform manual.

◆ ***WARNING!***

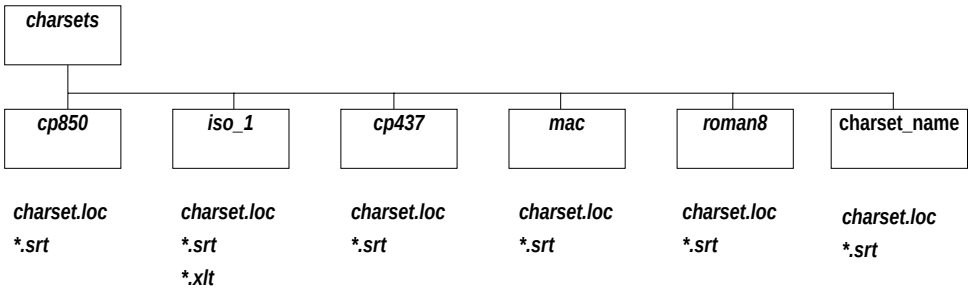
Do not alter any of the internationalization files. If you need to install a new terminal definition or sort order, contact your local Sybase distributor.

Character Sets Directory Structure

The following diagram shows the directory structure for the Western European character sets that come with Adaptive Server. There is a separate subdirectory for each character set in the *charsets* directory. Within the subdirectory for each character set (for example, *cp850*) are the character set and sort order definition files and terminal-specific files.

If you load additional character sets from the Sybase Character Sets product, they will also appear in the *charsets* directory.

Character Sets and Global Variables



The following global variables contain information about character sets:

@@char_convert	Contains 0 if character set conversion is not in effect. Contains 1 if character set conversion is in effect.
@@client_csname	The client's character set name. Set to NULL if client character set has never been initialized; otherwise, it contains the name of the most recently used character set.

@@client_csid	The client's character set ID. Set to -1 if client character set has never been initialized; otherwise, it contains the most recently used client character set ID from *syscharsets*.
@@maxcharlen	The maximum length, in bytes, of a character in Adaptive Server's default character set.
@@ncharsize	The average length, in bytes, of a national character.

Software Messages

By default, when you install Adaptive Server, U.S. English messages are loaded. International installations of Adaptive Server are supported with the Language Modules product that contain files with translated software messages and language or locale formats. These files, located in the locales subdirectory of the Adaptive Server installation directory, are called **localization files**.

Types of Localization Files

Several kinds of localization files are supplied with Adaptive Server and the Language Modules product, as shown in the following table.

Table 13-2: Localization files

File	Location	Purpose and Contents
locales.dat	In the *locales* directory	Used by client applications to identify the default message language and default character set.
server.loc	In the character set subdirectories under each language subdirectory in the *locales* directory	Software messages translated into the local language. Sybase products have product-specific *.*loc* files. If an entry is not translated, that software message or string appears in U.S. English instead of the local language.
common.loc	In each language and character set directory of the *locales* directory	Contains the local names of the months of the year and their abbreviations and information about the local date, time, and money formats.

◆ ***WARNING!***

Do not alter any of the localization files. If you need to alter any information in those files, contact your local Sybase distributor.

Software Messages Directory Structure

Figure 13-1 shows how localization files are arranged in the *locales* directory. Within the *locales* directory is a subdirectory for each language installed. By default, there is always a *us_english* subdirectory. (On PC platforms, this directory is called *english*.) If you install language modules for additional languages, you will see subdirectories for those languages. Within each language are subdirectories for the supported character sets; for example, *cp850* is a supported character set for us_english. Software message files for each of the different Sybase products reside in the subdirectory for each character set. During the Adaptive Server installation, when you are prompted to select the languages you want installed on Adaptive Server, the install program lists the supported software message languages.

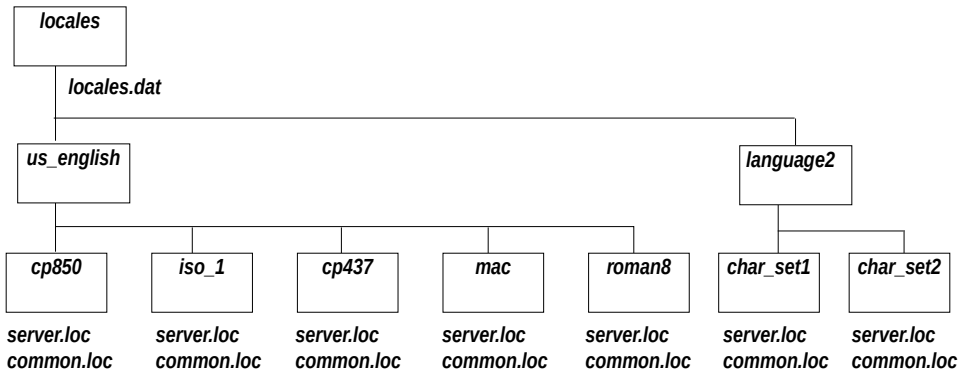


Figure 13-1: Messages directory structure

Message Languages and Global Variables

The following global variables contain information about languages:

@@langid	Contains the local language ID of the language currently in use (specified in *syslanguages.langid*).
@@language	Contains the name of the language currently in use (specified in *syslanguages.name*).

Disabling Character Set Conversion Between Adaptive Server and Clients

If a client uses a different character set from the one used by Adaptive Server, Adaptive Server's default is to convert the data to the server's character set when data is inserted or loaded and to convert back to the client's character set when data is returned to the user. This default behavior supports users of many different systems in a heterogeneous environment; however, in some cases, no conversion is needed.

For example, if some clients use Latin-1 (iso_1) and Adaptive Server uses Roman-8 (roman8) as its default character set, data from the clients is converted to Roman-8 when it is loaded into Adaptive Server. For clients using Latin-1, the data is reconverted when it is sent to the client; for clients using the same character set as Adaptive Server, the data is not converted. However, if all clients use Latin-1, especially if the data is copied to another Adaptive Server using Latin-1, the System Administrator may want to disable character set conversion entirely. All data is then stored in the client's character set in Adaptive Server.

The configuration parameter **disable character set conversions** controls character conversion server-wide. The default, 0, enables conversions. Client applications can request that no conversion take place.

Changing the Default Character Set, Sort Order, or Language

A System Administrator can change the character set, sort order, or message language used by Adaptive Server. Because a sort order is built on a specific character set, changing character sets always involves a change in sort order. However, you can change the sort order without changing character sets, because more than one sort order may be available for a character set.

This section summarizes the steps to take before and after changing Adaptive Server's character set, sort order, or message language. For procedures on how to configure the character set, sort order, or message language, see the configuration documentation for your platform.

Changing the Default Character Set

Adaptive Server can have only one **default character set**, the character set in which data is stored in its databases. When Adaptive Server is first installed, a default character set is specified.

◆ ***WARNING!***

Please read the following carefully and exercise extreme caution when changing the default character set in Adaptive Server.

When you change the default character set in Adaptive Server:

- You must convert any existing data to the new default character set:
  - First, copy the data out using **bcp**.
  - Then, change the default character set.
  - Then, use **bcp**, with the appropriate flags for data conversion, to copy the data back into the server.

  See the *Utility Programs* manual for your platform manual for more information about **bcp**.
- Code conversions must be supported for the character sets.

  Code conversion between the character set of the existing data and the new default character set must be supported. If it is not, conversion errors will occur and the data will not be converted correctly. See Chapter 14, "Configuring Client/Server Character Set Conversions," for more information about supported character set conversions.
- Some conversion errors may occur.

  Even if conversions are supported between the character sets, some conversion errors may occur because there are minor differences between the character sets, and some characters do not have equivalents in other character sets. Rows containing problematic data may not get copied back into the database or data may contain partial or invalid characters.

If your existing data is 7-bit ASCII, it does not require conversion to the new default character set. You can change the default without first copying your data out of the server.

Changing the Default Sort Order

Adaptive Server can have only one **default sort order**, the collating sequence it uses to order data. When you consider changing the sort order for character data on a particular Adaptive Server, keep this in mind: All of your organization's Adaptive Servers should have the same sort order. A single sort order enforces consistency and makes distributed processing easier to administer.

Changing the sort order may result in invalid indexes, and you may have to rebuild your indexes. For more information, see "If You Changed the Sort Order or Default Character Set" on page 13-10.

Getting Information About Sort Orders

The **sp_helpsort** system procedure displays Adaptive Server's default sort order, its character set, and a table of its primary sort orders. Its syntax is:

```
sp_helpsort
```

For more information about the different sort orders, see the configuration documentation for your platform.

Database Dumps and Configuration Changes

➤ ***Note***

Back up all databases in Adaptive Server both before and after you change character sets or sort orders.

In most instances, you cannot reload your data from a database dump after reconfiguring the Adaptive Server default character set and sort order.

Do not use a database dump if the following is true:

- If a database contains 8-bit character data, and you want the data to be converted to the new character set, do not load a database dump of the data into an Adaptive Server with the new default

character set. Adaptive Server interprets the data loaded as if it is in the new character set, and the data will be corrupted.

- If you are changing only the default sort order and not the default character set, you cannot load a database from a dump that was performed before you changed the sort order. If you attempt to do so, an error message appears, and the load is aborted.
- If you change the default character set, and either the old or the new sort order is not binary, you cannot load a database dump that was made before you changed the character set.

Instead, use **bcp** to copy the data out of and into your databases.

You can use a database dump if your data does not have to be converted to the new character set and both the old and the new character sets use binary sort order, you can restore your database from backups that were made before the character set was reconfigured.

The Steps Involved

Several steps are involved in changing the Adaptive Server sort order, message language, or character set.

Preliminary Steps

Before you run the installation program to reconfigure Adaptive Server:

1. Dump all user databases and the *master* database. If you have made changes to *model* or *sybsystemprocs*, dump them also.
2. Load the language module if it is not already loaded (see the configuration documentation for your platform for complete instructions).
3. If you are changing the Adaptive Server default character set, and your current databases contain non-7-bit data, use **bcp** to copy the existing data out of your databases.

At this point, you can run the Adaptive Server installation program to configure languages, character sets, or sort orders, as well as other options.

Steps to Configure Languages, Character Sets, and Sort Orders

When you install Adaptive Server, the installation program lets you:

- Install or remove message languages and character sets included with Adaptive Server
- Change the default message language or character set
- Select a different sort order

See the configuration documentation for your platform for instructions on using the installation program.

If you need to reconfigure the language, character set, or sort order, use the **sqlloc** utility, described in *Utility Programs for UNIX Platforms*. If you are using Windows NT, use the Server Config utility, described in *Configuring Adaptive Server for Windows NT*. If you are adding a new character set that is not included with Adaptive Server, see the *Sybase Character Sets* manual for complete instructions.

Final Steps

If you installed additional languages but did not change Adaptive Server's character set or sort order, you have completed the reconfiguration process.

If you changed the Adaptive Server default character set, and your current databases contain non-7-bit data, copy your data back into your databases, using **bcp** with the necessary flags to enable conversion.

If you changed Adaptive Server's default sort order or character set, also read "If You Changed the Sort Order or Default Character Set" on page 13-10.

Setting the User's Default Language

If an additional language is installed, users running client programs can run **sp_modifylogin** to set that language as their default language rather than Adaptive Server's default language.

If You Changed the Sort Order or Default Character Set

This section describes recovery after reconfiguration and the steps you may need to follow if you changed Adaptive Server's sort order or default character set.

If you changed sort orders, you need to do the following after reconfiguring Adaptive Server:

- Run **sp_indsuspect** to find user indexes that may no longer be valid.

- Rebuild suspect user indexes using the **dbcc reindex** command.

For more information, see "Using sp_indsuspect to Find Corrupt Indexes" on page 13-11, and"Rebuilding Indexes After Changing the Sort Order" on page 13-12.

If you changed to a multibyte character set from any other character set (either multibyte or single-byte), you must upgrade any existing *text* values with **dbcc fix_text**. See "Upgrading text Data After Changing Character Sets" on page 13-13 for more information.

Recovery After Reconfiguration

Every time Adaptive Server is stopped and restarted, recovery is performed automatically on each database. Automatic recovery is covered in detail in Chapter 20, "Developing a Backup and Recovery Plan."

After recovery is complete, the new sort order and character set definitions are loaded.

If the sort order has been changed, Adaptive Server switches to single-user mode to allow the necessary updates to system tables and to prevent other users from using the server. Each system table with a character-based index is automatically checked to see if any indexes have been corrupted by the sort order change. Character-based indexes in system tables are automatically rebuilt, if necessary, using the new sort order definition.

After the system indexes are rebuilt, character-based user indexes are marked "suspect" in the *sysindexes* system table, without being checked. User tables with suspect indexes are marked "read only" in *sysobjects* to prevent updates to these tables and use of the "suspect" indexes until they have been checked and, if necessary, rebuilt.

Next, the new sort order information replaces the old information in the area of the disk that holds configuration information. Adaptive Server then shuts down so that it starts for the next session with a "clean slate."

Using *sp_indsuspect* to Find Corrupt Indexes

After Adaptive Server shuts down, restart it, and use **sp_indsuspect** to find the user tables that need to be reindexed. The syntax is:

```
sp_indsuspect [tab_name]
```

where *tab_name* is the optional name of a specific table. If *tab_name* is missing, then **sp_indsuspect** creates a list of all tables in the current database that has indexes marked "suspect" when the sort order changes.

In this example, running **sp_indsuspect** in *mydb* database yields one suspect index:

```
sp_indsuspect

Suspect indexes in database mydb
Own.Tab.Ind (Obj_ID, Ind_ID) =
dbo.holdings.h_name_ix(160048003, 2)
```

Rebuilding Indexes After Changing the Sort Order

The **dbcc reindex** command checks the integrity of indexes on user tables by running a "fast" version of **dbcc checktable**. For details, see "dbcc checktable" on page 18-11. **dbcc reindex** drops and rebuilds the indexes where the sort order used is not consistent with the new sort order. When **dbcc reindex** discovers the first index-related error, it displays a message, and then rebuilds the inconsistent indexes. The System Administrator or table owner should run **dbcc reindex** after changing the sort order in Adaptive Server.

The syntax for **dbcc reindex** is:

```
dbcc reindex ({table_name | table_id})
```

Run this command on all tables listed by **sp_indsuspect** as containing suspect indexes. For example:

```
dbcc reindex(titles)

One or more indexes are corrupt. They will be
rebuilt.
```

In the preceding example, **dbcc reindex** discovers one or more suspect indexes in the table *titles*; it drops and re-creates the appropriate indexes.

If the indexes for a table are already correct, or if there are no indexes for the table, **dbcc reindex** does not rebuild any indexes. It displays a message instead. If a table is suspected of containing corrupt data, the command is aborted. If that happens, an error message instructs the user to run **dbcc checktable**.

When **dbcc reindex** finishes successfully, all "suspect" marks on the table's indexes are removed. The "read only" mark on the table is also removed, and the table can be updated. These marks are removed whether or not any indexes have to be rebuilt.

dbcc reindex does not reindex system tables. System indexes are checked and rebuilt, if necessary, as an automatic part of recovery after Adaptive Server is restarted following a sort order change.

Upgrading *text* Data After Changing Character Sets

The **dbcc fix_text** command upgrades *text* values after an Adaptive Server's character set has been changed to a **multibyte character set**.

The syntax for **dbcc fix_text** is:

```
dbcc fix_text ({table_name | table_id})
```

Changing to a multibyte character set makes the management of *text* data more complicated. A *text* value can be large enough to cover several pages; therefore, Adaptive Server must be able to handle characters that span page boundaries. To do so, Adaptive Server requires additional information on each of the *text* pages. The System Administrator or table owner must run **dbcc fix_text** on each table that has *text* data to calculate the new values needed.

➤ ***Note***

You must run **dbcc fix_text** if you are changing to a new multibyte character set from either a single-byte or a multibyte character set. You need to run **dbcc fix_text** only on tables that contain text data.

To see the names of all tables that contain *text* data, use this query:

```
select sysobjects.name
from sysobjects, syscolumns
where syscolumns.type = 35
and sysobjects.id = syscolumns.id
```

The System Administrator or table owner must run **dbcc fix_text** to calculate the new values needed.

The syntax of **dbcc fix_text** is:

```
dbcc fix_text (table_name | table_id)
```

The table named must be in the current database.

dbcc fix_text opens the specified table, calculates the character statistics required for each *text* value, and adds the statistics to the appropriate page header fields. This process can take a long time, depending on the number and size of the *text* values in a table. **dbcc fix_text** can generate a large number of log records, which may fill up the transaction log. **dbcc fix_text** performs updates in a series of small

transactions so that if a log becomes full, only a small amount of work is lost.

If you run out of log space, clear out your log (see Chapter 21, "Backing Up and Restoring User Databases"). Then, restart **dbcc fix_text**, using the same table that was being upgraded when the original **dbcc fix_text** halted. Each multibyte text value contains information that indicates whether it has been upgraded, so **dbcc fix_text** upgrades only the *text* values that were not processed in earlier passes.

If your database stores its log on a separate segment, you can use thresholds to manage clearing the log. See Chapter 23, "Managing Free Space with Thresholds."

If **dbcc fix_text** does not complete updating a table, an error message such as the following appears:

```
Not all of the TEXT pages in tablename have been
successfully updated, however, dbcc fix_text is
restartable. Please issue the command again once
any other errors have been addressed.
```

If **dbcc fix_text** cannot acquire a needed lock on a text page, it reports the problem and continues with the work, like this:

```
Unable to acquire an exclusive lock on text page
408. This text value has not been recalculated.
In order to recalculate those TEXT pages you must
release the lock and reissue the dbcc fix_text
command.
```

Retrieving *text* Values After Changing Character Sets

If you attempt to retrieve *text* values after changing to a multibyte character set, and you have not run **dbcc fix_text**, the command fails and this error message is generated:

```
Adaptive Server is now running a multi-byte
character set, and this TEXT column's character
counts have not been recalculated using this
character set. Use dbcc fix_text before running
this query again.
```

Installing Date Strings for Unsupported Languages

You can use **sp_addlanguage** to install names for the days of the week and months of the year for languages that do not have language modules. With **sp_addlanguage**, you define:

- A language name and (optionally) an alias for the name
- A list of the full names of months and a list of abbreviations for the month names
- A list of the full names of the days of the week
- The date format for entering dates (such as month / day / year)
- The number of the first day of the week

This example adds the information for Italian:

```
sp_addlanguage italian, italiano,
"gennaio,febbraio,marzo,aprile,maggio,giugno,luglio,agosto,settem
bre,ottobre,novembre,dicembre",
"genn,feb,mar,apr,mag,giu,lug,ago,sett,ott,nov,dic",
"lunedi,martedi,mercoledi,giovedi,venerdi,sabato,domenica",
dmy, 1
```

sp_addlanguage enforces strict data entry rules. The lists of month names, month abbreviations, and days of the week must be comma-separated lists with no spaces or line feeds (returns). Also, they must contain the right number of elements (12 for month strings, 7 for day-of-the-week strings.)

Valid values for the date formats are: *mdy*, *dmy*, *ymd*, *ydm*, *myd*, and *dym*. The *dmy* value indicates that the dates are in day / month / year order. This format affects only data entry; to change output format, you must use the **convert** function.

Server vs. Client Date Interpretation

Generally, date values are resolved on the client. When a user selects date values, Adaptive Server sends them to the client in internal format. The client uses the *common.loc* file and other localization files in the default language subdirectory of the *locales* directory on the client to convert the internal format to character data. For example, if the user's default language is Spanish, Adaptive Server looks for the *common.loc* file in the */locales/spanish/char_set* directory. It uses the information in the *common.loc* file to display, for example, *12 febrero 1997*.

Assume that the user's default language is set to Italian, a language for which Adaptive Server does not provide a language module, and that the date values in Italian have been added with **sp_addlanguage**. When the client connects to the server and looks for the *common.loc* file for Italian, it will not find the file. The client prints an error message and connects to the server. If the user then selects date values, the dates are displayed using the date strings for the server's default language. To display the date values added with **sp_addlanguage**, use the **convert** function to force the dates to be converted to character data at the server.

The following query generates a result set with the dates in the Adaptive Server default language:

```
select pubdate from titles
```

whereas the query below returns the date with the month names in Italian:

```
select convert(char(19),pubdate) from titles
```

14 Configuring Client / Server Character Set Conversions

This chapter describes how to configure character set conversion for situations in which the client uses a different character set from the one used by Adaptive Server. Topics include:

- Character Set Conversion in Adaptive Server 14-1
- Conversion Paths Supported 14-1
- Error Handling in Character Set Conversion 14-3
- Setting Up the Conversion Process 14-4
- Display and File Character Set Command Line Options 14-7

Character Set Conversion in Adaptive Server

Clients that use different character encoding schemes can connect over a network to the same Adaptive Server. In a Western European setting, for example, a server that runs in an ISO 8859-1 (iso_1) environment may be connected to a client that runs in a Roman 8 (roman8) environment. In Japan, a server that runs in an EUC JIS (eucjis) environment may be connected to a client that runs in a Shift-JIS (sjis) environment. This chapter describes the character set conversion features of Adaptive Server and the utilities **isql**, **bcp**, and **defncopy**.

Conversion Paths Supported

Adaptive Server supports **character set conversion** among the character sets within the language groups shown in Table 14-1. Character set names in parentheses are the names used by Adaptive Server.

An additional character set, ASCII 7 (ascii_7), is compatible with all character sets. If either the Adaptive Server or the client's character

set is ascii_7, any 7-bit ASCII character can pass between the client and server unaltered. Other characters produce conversion errors.

Table 14-1: Supported character set conversions

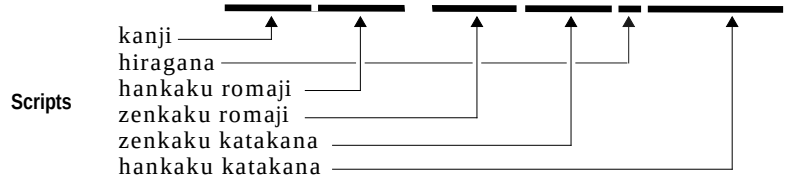
Language Group	Character Sets
Cyrllic Script	Code Page 1251 (cp1251), Code Page 855 (cp855), Code Page 866 (cp866), ISO 8859-5 (iso88595), Koi 8 (koi8), Maccyr (mac_cyr)
Eastern European	Code Page 1250 (cp1250), Code Page 852 (cp852), ISO 8859-2 (iso88592), Macee (mac_ee)
Greek	Code Page 1253 (cp1253), Code Page 869 (cp869), Greek 8 (greek8), ISO8859-7 (iso88597), Macgrk2 (macgrk2)
Japanese	DEC-Kanji (deckanji), EUC-JIS (eucjis), Shift-JIS (sjis)
Turkish	Code Page 1254 (cp1254), Code Page 857 (cp857), ISO 8859-9 (iso88599), Macturk (macturk), Turkish 8 (turkish8)
Western European	ASCII 8 (ascii_8), Code Page 437 (cp437), Code Page 850 (cp850), ISO 8859-1 (iso_1), Mac (mac), Roman 8 (roman8)
Arabic, Chinese (Simplified and Traditional), or Hebrew	No code set conversion is supported in these language groups.

Characters That Cannot Be Converted

In converting one character set to another, some characters may not be converted. Here are two possibilities:

- The character exists (is encoded) in the source character set, but it does not exist in the target character set. For example, the character "Œ," the OE ligature, is part of the Macintosh character set (code point 0xCE). This character does not exist in the ISO 8859-1 character set. If "Œ" exists in data that is being converted from the Macintosh to the ISO 8859-1 character set, it causes a conversion error.
- The character exists in both the source and the target character set, but in the target character set, the character is represented by a different number of bytes than in the source character set. Figure 14-1 compares the EUC JIS and Shift-JIS encodings for the same sequence of characters in a Japanese environment. Kanji, Hiragana, Hankaku Romaji, Zenkaku Romaji, and Zenkaku

Katakana characters are represented by the same number of bytes in both character sets and can be converted between EUC JIS and Shift-JIS. However, Hankaku Katakana characters (the last set of characters in the example) are represented by two bytes in EUC JIS and by a single byte in Shift-JIS. These characters cannot be converted.



		conversion	
	EUC JIS form of use		**Shift-JIS form of use**
Numeric Repre-sentation	c6fc CPT b8ec 53 59 42 41 53 45 20 a3d3 a3d1 a3cc a5b5 a1bc a5d9 a4ce 8ec3 8ede 8eb0 8ec0 8ecd 8ede 8eb0 8ebd	→	93fa 967b 8cea 53 59 42 41 53 45 20 8272 8270 826b 8354 815b 8378 82cc c3 de b0 c0 cd de b0 bd

Figure 14-1: Comparison of EUC JIS and Shift-JIS encoding for Japanese characters

In addition to Hankaku Katakana, user-defined characters (Gaiji) cannot be converted between Japanese character sets.

Error Handling in Character Set Conversion

Adaptive Server's character set conversion filters report conversion errors when a character exists in the client's character set but not in the server's character set, or vice versa. Adaptive Server must guarantee that data successfully converted on input to the server can be successfully converted back to the client's character set when the client retrieves that data. To do this effectively, Adaptive Server must avoid putting suspect data into the database.

When Adaptive Server encounters a conversion error in the data being entered, it generates this error message:

```
Msg 2402, Severity 16 (EX_USER):
Error converting client characters into server's
character set. Some character(s) could not be
converted.
```

A conversion error prevents query execution.

When Adaptive Server encounters a conversion error while sending data to the client, it replaces the bytes of the suspect characters with ASCII question marks (?). However, the query batch continues to completion. When the statement is complete, Adaptive Server sends the following message:

```
Msg 2403, Severity 16 (EX_INFO):
WARNING! Some character(s) could not be converted
into client's character set. Unconverted bytes
were changed to question marks ('?').
```

See "Controlling Character Conversion During a Session" on page 14-6 to learn how to turn off error reporting for data being sent from server to client.

Setting Up the Conversion Process

Character set conversion begins at login or when the client requests conversion with the **set char_convert** command during a work session. If the client is using Open Client DB-Library release 4.6 or later, and the client and Adaptive Server are using different character sets, conversion is turned on during the login process and is set to a default, based on the character set that the client is using. Character set conversion can be controlled in the stand-alone utilities **isql**, **bcp**, and **defncopy** with a command line option. See "Display and File Character Set Command Line Options" on page 14-7 for details.

When a client requests a connection, Adaptive Server determines whether it can convert from the client's character set to its own character set. If it can, it sets up the appropriate character set conversion filters so that any character data read from or sent to the client automatically passes through them. Next, Adaptive Server checks the user name and password. These have already been read and must be converted. If they cannot be converted, the login is denied. If conversion on the name and password succeeds, Adaptive Server looks up the converted strings in *syslogins*.

If Adaptive Server cannot perform the requested conversions, it sends an error message to the client. The initial message explains why conversion cannot be done and is followed by this message:

```
Msg #2411, Severity 10 (EX_INFO):
No conversions will be done.
```

Next, Adaptive Server tries to find the user name and password in their unconverted form in *syslogins*. If it cannot find them, the login

is denied. If it succeeds, the login is granted, but no character set conversion takes place.

➤ ***Note***

Machine names, user names, and passwords in heterogeneous environments should be composed entirely of 7-bit ASCII characters. If the client's request for character set conversion fails, the login still succeeds, if Adaptive Server finds the unconverted user name and password in *syslogins*.

If the request for conversion fails, or if the client's character set is set to ascii_7, the language for the session is forced to us_english. If the user had requested a different language, an informational message would appear, stating that the language for the session was being forced to us_english.

Specifying the Character Set for Utility Programs

A command line option for the **isql**, **bcp**, and **defncopy** utilities specifies the client's character set.

Here are the choices:

- **-J** *charset_name (*UNIX and PC*)* or **/clientcharset** = *charset_name* (OpenVMS) sets the client's character set to the *charset_name*.
- **-J** or **/clientcharset** with no character set name sets the client's character set to NULL. No conversion takes place, and no message is sent.

Omitting the client character set's command line flag sets the character set to a default for the platform. This default may not be the character set that the client is using. See the *Utility Programs* manual for your platform manual for information on the default for your platform.

➤ ***Note***

The **-J** flag is used differently by pre-4.9 SQL Server running on OS/2.

Controlling Character Conversion During a Session

The **set char_convert** command determines how character set conversion operates during a particular work session. Use **set char_convert** to:

- Set character set conversion to "on" or "off"
- Start conversion to a specific character set
- Set error reporting to "on" or "off"

The syntax for **set char_convert** is:

```
set char_convert {off |
                 {on [with {error | no_error}]} |
                 charset [with {error | no_error}]}
```

Depending on the arguments, the command:

- Turns character set conversion on and off between Adaptive Server and a client, when used with **off** or **on**:

  ```
  set char_convert off
  ```

  set char_convert off turns conversion off so that characters are sent and received unchanged. **set char_convert on** turns conversion back on after it was turned off. If character set conversion was not turned on during the login process or by the **set char_convert** *charset* command, then **set char_convert on** generates an error message.

- Turns off the printing of error messages when the **with no_error** option is included. When you use **with no_error**, Adaptive Server does not notify the application when characters from Adaptive Server cannot be converted to the client's character set. Error reporting is initially set to "on" when a client connects with Adaptive Server. If you do not want error reporting, you must turn it off for each session. To turn error reporting back on within a session, use **set char_convert on with error.**

  Whether or not error reporting is turned on, the bytes that cannot be converted are replaced with ASCII question marks (?).

- Starts conversion between the server character set and a different client character set, when used with a *charset* value. *charset* can be either the character set's *id* or its *name* from *syscharsets*:

  ```
  set char_convert "cp850"
  ```

  If you request character set conversion with **set char_convert** *charset*, and Adaptive Server cannot perform the requested conversion, the conversion state remains the same as it was

before the request. For example, if character set conversion is off prior to the **set char_convert** *charset* command, conversion remains off if the request fails.

If the user is using a language other than us_english before entering this command:

```
set char_convert "ascii_7"
```

the language for the session is forced to us_english and an informational message appears. No message appears if the session is already in us_english.

Display and File Character Set Command Line Options

Although the focus of this chapter is on character set conversion between client and Adaptive Server, character set conversion may be needed in two other places:

- Between the client and a terminal
- Between the client and a file system.

Figure 14-2 illustrates the paths and command line options that are available in the stand-alone utilities **isql**, **bcp**, and **defncopy.**

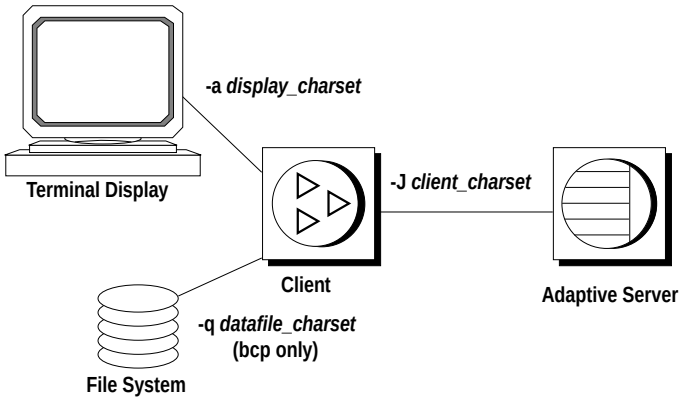


Figure 14-2: Where character set conversion may be needed

As described earlier, the **-J** or **/clientcharset** command line option specifies the character set used by the client when it sends and receives character data to and from Adaptive Server.

Setting the Display Character Set

Use the **-a** command line option (**/dispcharset** on OpenVMS) if you are running the client from a terminal with a character set that differs from the client character set. In Figure 14-2, the **-a** option and the **-J** option are used together to identify the character set translation file (*.xlt* file) needed for the conversion.

Use the **-a** command line option (**/dispcharset** on OpenVMS) without the **-J** option (**/clientcharset** on OpenVMS) only if the client character set is the same as the default character set.

Setting the File Character Set

Use the **-q** command line option (**/filecharset** on OpenVMS) if you are running **bcp** to copy character data to or from a file system that uses a character set that differs from the client character set. In Figure 14-2, use the **-q** or **/filecharset** option and the **-J** or **/clientcharset** option together to identify the character set translation file (*.xlt* file) needed for the conversion.

Managing Databases and Database Objects

15 Creating and Managing User Databases

This chapter explains how to create and manage user databases. Topics include:

- Commands for Creating and Managing User Databases 15-1
- Using the create database Command 15-3
- Assigning Space and Devices to Databases 15-5
- Placing the Transaction Log on a Separate Device 15-7
- Using the for load Option for Database Recovery 15-10
- Using the with override Option with create database 15-11
- Changing Database Ownership 15-12
- Using the drop database Command 15-14
- System Tables That Manage Space Allocation 15-15
- Getting Information About Database Storage 15-18

Commands for Creating and Managing User Databases

Table 15-1 summarizes the commands for creating, modifying, and dropping user databases and their transaction logs.

Table 15-1: Commands for managing user databases

Command	Task
create database...on *dev_name* or **alter database...on** *dev_name*	Makes database devices available to a particular Adaptive Server database. The **log on** clause to **create database** places the database's logs on a particular database device.
create database... or **alter database...**	When used without the **on** *dev_name* clause, these commands allocate space from the default pool of database devices.
dbcc checktable(syslogs)	Reports the size of the log.
sp_logdevice	Specifies a device that will store the log when the current log device becomes full.
sp_helpdb	Reports information about a database's size and devices.

Table 15-1: Commands for managing user databases (continued)

Command	Task
sp_spaceused	Reports a summary of the amount of storage space used by a database.

Permissions for Managing User Databases

By default, only the System Administrator has **create database** permission. The System Administrator can grant permission to use the **create database** command. However, in many installations, the System Administrator maintains a monopoly on **create database** permission in order to centralize control of database placement and database device allocation. In these situations, the System Administrator creates new databases on behalf of other users and then transfers ownership to the appropriate user(s).

To create a database and transfer ownership to another user, the System Administrator:

1. Issues the **create database** command
2. Switches to the new database with the **use** *database* command
3. Executes the system procedure **sp_changedbowner**, as described in "Changing Database Ownership" on page 15-12.

The System Administrator can also grant permission to create databases. The user that receives **create database** permission must also be a valid user of the *master* database, since all databases are created while using *master*.

The fact that System Administrators seem to operate outside the protection system serves as a safety precaution. For example, if a Database Owner forgets his or her password or accidentally deletes all entries in *sysusers*, a System Administrator can repair the damage using the backups or dumps that are made regularly.

Permission to use the **alter database** or **drop database** command defaults to the Database Owner, and permission is automatically transferred with database ownership. **alter database** and **drop database** permission cannot be changed with **grant** or **revoke**.

Using the *create database* Command

Use the **create database** command to create user databases. You must have **create database** permission, and you must be a valid user of *master* (added with **sp_adduser**). Always type the **use master** command before creating a new database.

➤ ***Note***

Each time you enter the **create database** command, dump the *master* database. This makes recovery easier and safer in case *master* is later damaged. See Chapter 22, "Restoring the System Databases," for more information.

create database Syntax

The **create database** syntax is:

```
create database database_name
   [on {default | database_device} [= size]
       [, database_device [= size]...]
   [log on database_device [ = size ]
       [, database_device [= size]]...]
   [with override]
   [for load]
```

A database name must follow the rules for identifiers. You can create only one database at a time.

In its simplest form, **create database** creates a database on the default database devices listed in *master..sysdevices*:

```
create database newpubs
```

You can control different characteristics of the new database by using the **create database** clauses:

- The **on** clause specifies the names of one or more database devices and the space allocation, in megabytes, for each database device. See "Assigning Space and Devices to Databases" on page 15-5 for more information.
- The **log on** clause places the **transaction log** (the *syslogs* table) on a separate database device with the specified or default size. See "Placing the Transaction Log on a Separate Device" on page 15-7 for more information.

- The **for load** option causes Adaptive Server to skip the page-clearing step during database creation. You can use this clause if you intend to load a dump into the new database as the next step. See "Using the for load Option for Database Recovery" on page 15-10 for more information.
- The **with override** option allows Adaptive Servers on machines with limited space to maintain their logs on separate device fragments from their data. Use this option **only** when you are putting log and data on the same logical device. See "Using the with override Option with create database" on page 15-11 for more information.

How *create database* Works

When a user with the required permission issues the **create database** statement, Adaptive Server:

- Verifies that the database name specified in the statement is unique.
- Makes sure that the database device names specified in the statement are available.
- Finds an unused identification number for the new database.
- Assigns space to the database on the specified database devices and updates *master..sysusages* to reflect these assignments.
- Inserts a row into *sysdatabases.*
- Makes a copy of the *model* database in the new database space, thereby creating the new database's system tables.
- Clears all the remaining pages in the database device. If you are creating a database in order to load a database dump, the **for load** option skips page clearing (the page clearing step is performed after the load completes).

The new database initially contains a set of system tables with entries that describe the system tables themselves. The new database inherits all the changes you have made to the *model* database. These can include:

- The addition of user names.
- The addition of objects.
- The database option settings. Originally, the options are set to "off" in *model*. If you want all of your new user databases to inherit particular options, change the options in *model* with the system procedure **sp_dboption**. See Chapter 2, "System Databases,"

for more information about *model*. See Chapter 16, "Setting Database Options," for more information about changing database options.

Adding Users to Databases

After creating a new database, the System Administrator or Database Owner can manually add users to the database with **sp_adduser**. This task can be done with the help of the System Security Officer, if new Adaptive Server logins are required. See the *Security Administration Guide* for details on managing Adaptive Server logins and database users.

Assigning Space and Devices to Databases

Adaptive Server allocates storage space to databases when a user enters the **create database** or **alter database** command. The **create database** command can specify one or more database devices, along with the amount of space on each that is to be allocated to the new database.

➤ ***Note***

Also use the **log on** clause to place a production database's transaction log on a separate device. See "Placing the Transaction Log on a Separate Device" on page 15-7 for more information.

If you use the **default** keyword, or if you omit the **on** clause altogether, Adaptive Server puts the database on one or more of the default database devices specified in *master..sysdevices*. See "Designating Default Devices" on page 6-7 for more information about the default pool of devices.

To specify a size (4MB in the following example) for a database that is to be stored in a default location, use **on default = size** like this:

```
create database newpubs
on default = 4
```

To place the database on specific database devices, give the name(s) of the database device(s) on which you want it stored. As the syntax indicates, you can request that a database be stored on more than one database device, with a different amount of space on each. All the database devices named in **create database** must be listed in *sysdevices*. In other words, they must have been initialized with **disk init**. See

Chapter 6, "Initializing Database Devices," for instructions about using **disk init**.

The following statement creates the *newdb* database and allocates 3MB on *mydata* and 2MB on *newdata*. As in the preceding example, the database and transaction log are not separated:

```
create database newdb
on mydata = 3, newdata = 2
```

◆ ***WARNING!***

Unless you are creating a small or noncritical database, always place the log on a separate database device. Follow the instructions under "Placing the Transaction Log on a Separate Device" on page 15-7 to create production databases.

If the amount of space you request on a specific database device is unavailable, Adaptive Server creates the database with as much space as possible on each device. Then, Adaptive Server displays a message informing you how much space it has allocated on each database device. (This is not considered an error.) If there is less than the minimum space necessary for a database on the specified database device (or on the default, if you do not specify a name), the **create database** command fails.

Default Database Size and Devices

If you omit the size parameter in the **on** clause, Adaptive Server creates the database with a default amount of space. This amount is the larger of the sizes specified by the **default database size** configuration parameter and the *model* database.

The size of *model* and the value of **default database size** are initially set to 2MB. To change the size of *model*, allocate more space to it with **alter database**. To change the **default database size** configuration parameter, use **sp_configure**. Changing **default database size** enables you to set the default size for new databases to any size between 2MB and 10,000MB. See "default database size" on page 11-101 for complete instructions.

If you omit the **on** clause completely, the size of the database is the default size, as described above. The space is allocated, in alphabetical order by database device name, from the default database devices specified in *master..sysdevices*.

Use the following query to see the logical names of default database devices:

```
select name
    from sysdevices
    where status & 1 = 1
    order by name
```

sp_helpdevice also displays "default disk" as part of the description of database devices.

Estimating the Required Space

The allocation decisions you make when you issue **create database** or **alter database** are important, because it is difficult to reclaim storage space once it has been assigned. You can always add space; however, you cannot deallocate space that has been assigned to a database, unless you drop the database first.

You can estimate the size of the tables and indexes for your database by using the **sp_estspace** system procedure or by calculating the value. See Chapter 6, "Determining or Estimating the Sizes of Tables and Indexes," in the *Performance and Tuning Guide* for instructions about estimating the size of objects.

Placing the Transaction Log on a Separate Device

The **log on** clause to **create database** places the **transaction log** (the *syslogs* table) on a separate database device. Placing the logs on a separate database device:

- Lets you use the **dump transaction** command, rather than **dump database**, thus saving time and tapes.
- Lets you establish a fixed size for the log to keep it from competing for space with other database activity.
- Creates default free-space threshold monitoring on the log segment and allows you to create additional free-space monitoring on the log and data portions of the database. See Chapter 23, "Managing Free Space with Thresholds," for more information.
- Improves performance.
- Ensures full recovery from hard disk crashes. A special argument to **dump transaction** lets you dump your transaction log, even when your data device is on a damaged disk.

Unless you are creating very small, noncritical databases, always place the log on a separate database device.

➤ ***Note***

If the log and its database share the same device, subsequent use of the system procedure **sp_logdevice** affects only future writes to the log, as described in "Moving the Transaction Log to Another Device" on page 15-9. It does not immediately move the first few log pages that were written when the database was created. This creates exposure problems in certain recovery situations, and is not recommended.

To specify a size and device for the transaction log, use the **log on *device* = *size*** clause to **create database**. For example, the following statement creates the *newdb* database, allocates 8MB on *mydata* and 4MB on *newdata*, and places a 3MB transaction log on a third database device, *tranlog*:

```
create database newdb
on mydata = 8, newdata = 4
log on tranlog = 3
```

Estimating the Transaction Log Size

Two factors determine the size of the transaction log:

- The amount of update activity in the associated database
- The frequency of transaction log dumps

This is true whether you perform transaction log dumps manually or use threshold procedures to automate the task. As a general rule, allocate to the log 10 to 25 percent of the space that you allocate to the database.

Inserts, deletes, and updates increase the size of the log. **dump transaction** decreases its size by writing committed transactions to disk and removing them from the log. Since **update** statements require logging both the "before" and "after" images of a row, applications that update many rows at once should plan on the transaction log being at least twice as large as the number of rows to be updated at the same time, or twice as large as your largest table. Or you can **batch** the updates in smaller groups, performing transaction dumps between the batches.

In databases that have a lot of insert and update activity, logs can grow very quickly. To determine the required log size, periodically

check the size of the log. This will also help you in choosing thresholds for the log and in scheduling the timing of transaction log dumps. To check the space used by a database's transaction log, first use the database. Then, enter the command:

```
dbcc checktable(syslogs)
```

dbcc reports the number of data pages being used by the log. If your log is on a separate device, **dbcc checktable** also tells you how much space is used and how much is free. Here is sample output for a 2MB log:

```
Checking syslogs
The total number of data pages in this table is 199.
*** NOTICE: Space used on the log segment is 0.39 Mbytes, 19.43%.
*** NOTICE: Space free on the log segment is 1.61 Mbytes, 80.57%.
Table has 1661 data rows.
```

You can also use the following Transact-SQL statement to check on the growth of the log:

```
select count(*) from syslogs
```

Repeat either command periodically to see how fast the log grows.

Default Log Size and Device

If you omit the *size* parameter in the **log on** clause, Adaptive Server allocates 2MB of storage on the specified log device. If you omit the **log on** clause entirely, Adaptive Server places the 2MB transaction log on the same database device as the data tables.

Moving the Transaction Log to Another Device

If you did not use the **log on** clause to **create database**, follow the instructions in this section to move your transaction log to another database device.

The system procedure **sp_logdevice** moves the transaction log of a database with log and data on the same device to a separate database device.. However, the transaction log remains on the original device until the allocated page has been filled and the transaction log has been dumped.

The syntax for **sp_logdevice** is:

```
sp_logdevice database_name, devname
```

The database device you name must be initialized with **disk init** and must be allocated to the database with **create** or **alter database**.

To move the entire transaction log to another device, complete these steps:

1. Execute **sp_logdevice**, naming the new database device.
2. Execute enough transactions to fill the page that is currently in use. Since a page contains 2048 bytes, you may need to update at least 2048 bytes. You can execute **dbcc checktable(syslogs)** before and after you start updating to determine when a new page is used.
3. Wait for all currently active transactions to finish to ensure that there are no active transactions on the database device. You may want to perform this entire activity after putting the database into single-user mode with **sp_dboption**.
4. Run **dump transaction**. The **dump transaction** command removes all the log pages that it writes to disk. As long as there are no active transactions in the part of the log on the old device, all of those pages will be removed. See Chapter 20, "Developing a Backup and Recovery Plan," for more information.
5. Run the system procedure **sp_helplog** to ensure that the complete log is on the new log device.

➤ ***Note***

When you move a transaction log, the space no longer used by the transaction log becomes available for data. However, you cannot reduce the amount of space allocated to a device by moving the transaction log.

Transaction logs are discussed in detail in Chapter 20, "Developing a Backup and Recovery Plan."

Using the *for load* Option for Database Recovery

Adaptive Server generally clears all unused pages in the database device when you create a new database. Clearing the pages can take several seconds or several minutes to complete, depending on the size of the database and the speed of your system.

Use the **for load** option if you are going to use the database for loading from a database dump, either for recovery from media failure or for moving a database from one machine to another. Using **for load** runs a

streamlined version of **create database** that skips the page clearing step. This creates a target database that can be used **only** for loading a dump.

If you create a database using the **for load** option, you can run only the following commands in the new database before loading a database dump:

- **alter database...for load**
- **drop database**
- **load database**

When you load a database dump, the new database device allocations for the database need to match the usage allocations in the dumped database. See Chapter 21, "Backing Up and Restoring User Databases," for a discussion of duplicating space allocation.

After you load the database dump into the new database, there are no restrictions on the commands you can use.

Using the *with override* Option with *create database*

This option allows Adaptive Server on machines that have limited space to maintain their logs on separate device fragments from their data. Use this option **only** when you put log and data on the same logical device. Although this is not recommended practice, it may be the only option available on machines with limited storage, especially if you need to get databases back online following a hard disk crash.

You will still be able to dump your transaction log, but if you experience a media failure, you will not be able to access the current log, since it is on the same device as the data. You will be able to recover only to the last transaction log dump, and all transactions between that point and the failure time will be lost.

In the following example, the log and data are on separate fragments of the same logical device:

```
create database littledb
    on diskdev1 = 4
    log on diskdev1 = 1
    with override
```

Use the **with override** option only if you are not concerned with up-to-the-minute recoverability.

Changing Database Ownership

A System Administrator might want to create the user databases and give ownership of them to another user after completing some of the initial work. The system procedure **sp_changedbowner** changes the ownership of a database. The procedure must be executed by the System Administrator in the database where the ownership will be changed. The syntax is:

```
sp_changedbowner loginame [, true ]
```

The following example makes the user "albert" the owner of the current database and drops the aliases of users who could act as the former "dbo":

```
sp_changedbowner albert
```

The new owner must already have a login name in Adaptive Server, but he or she cannot be a user of the database or have an alias in the database. You may have to use **sp_dropuser** or **sp_dropalias** before you can change a database's ownership. See the *Security Administration Guide* for more information about changing ownership.

To transfer aliases and their permissions to the new Database Owner, add the second parameter, **true**.

➤ ***Note***

You cannot change the ownership of the *master* database. It is always owned by the "sa" login.

Using the *alter database* Command

When your database or transaction log grows to fill all the space allocated with **create database**, you can use the **alter database** command to add storage. You can add space for database objects or the transaction log, or both. You can also use **alter database** to prepare to load a database from backup.

Permission to use the **alter database** command defaults to the Database Owner, and permission is automatically transferred with database ownership. For more information, see "Changing Database Ownership" on page 15-12. **alter database** permission cannot be changed with **grant** or **revoke**.

alter database Syntax

To extend a database, and to specify where storage space is to be added, use the full **alter database** syntax:

```
alter database database_name
     [on {default | database_device} [= size]
         [, database_device [= size]]...]
     [log on  {default | database_device} [= size]
         [, database_device [= size]]...]
     [with override]
     [for load]
```

In its simplest form, **alter database** adds 1MB from the default database devices. If your database separates log and data, the space you add is used only for data. Use **sp_helpdevice** to find names of database devices that are in your default list.

To add 1MB from a default database device to the *newpubs* database, enter:

```
alter database newpubs
```

The **on** and **log on** clauses in the **alter database** command operate like the corresponding clauses in the **create database** command. You can specify space on a default database device or some other database device, and you can name more than one database device. If you use **alter database** to extend the *master* database, you can extend it only on the master device. The minimum increase you can specify is 1MB (512 2K pages).

To add 3MB to the space allocated for the *newpubs* database on the database device named *pubsdata1*, enter:

```
alter database newpubs
on pubsdata1 = 3
```

If Adaptive Server cannot allocate the requested size, it allocates as much as it can on each database device, with a minimum allocation of .5MB (256 2K pages) per device. When **alter database** completes, it prints messages telling you how much space it allocated; for example:

```
Extending database by 1536 pages on disk pubsdata1
```

Check all messages to make sure the requested amount of space was added.

The following command adds 2MB to the space allocated for *newpubs* on *pubsdata1*, 3MB on a new device, *pubsdata2*, and 1MB for the log on *tranlog*:

```
alter database newpubs
on pubsdata1 = 2, pubsdata2 = 3
log on tranlog
```

➤ *Note*

Each time you issue the **alter database** command, dump the *master* database.

Using the *with override* Clause

Use the **with override** clause with **alter database** to create a device fragment containing log space on a device that already contains data or a data fragment on a device already in use for the log. Use this option only when you have no other storage options and when up-to-the-minute recoverability is not critical.

Using the *for load* Clause

Use the **for load** clause with **alter database** only after using **create database for load** to re-create the space allocation of the database being loaded into the new database from a dump. See Chapter 21, "Backing Up and Restoring User Databases," for a discussion of duplicating space allocation when loading a dump into a new database.

Using the *drop database* Command

Use the **drop database** command to remove a database from Adaptive Server, thus deleting the database and all the objects in it. This command:

- Frees the storage space allocated for the database
- Deletes references to the database from the system tables in the *master* database

Only the Database Owner can drop a database. You must be in the *master* database to drop a database. You cannot drop a database that is open for reading or writing by a user.

drop database Syntax

The syntax of this command is:

```
drop database database_name [, database_name]...
```

You can drop more than one database in a single statement. For example:

```
drop database newpubs, newdb
```

You must drop all databases from a database device before you can drop the database device itself. The command to drop a device is **sp_dropdevice**.

After you drop a database, dump the *master* database to ensure recovery in case *master* is damaged.

System Tables That Manage Space Allocation

For the user, creating a database on a database device and allocating a certain amount of space to it is just a matter of issuing a command. For Adaptive Server, the task is more complex.

First, Adaptive Server makes an entry for the new database in *sysdatabases*. Then, it checks *master..sysdevices* to make sure that the device names specified in the **create database** command actually exist and are database devices. If you did not specify database devices, or used the **default** option, Adaptive Server checks *master..sysdevices* and *master..sysusages* for free space on all devices that can be used for default storage. It performs this check in alphabetical order by device name.

The storage space from which Adaptive Server gathers the specified amount of storage need not be contiguous, and it can be extracted from whatever free space is available. The database storage space can even be drawn from more than one database device. Of course, a database is treated as a logical unit, even if it is stored on more than one database device.

Each piece of storage for a database must be at least 1 allocation unit—1/2MB, or 256 contiguous 2K pages. The first page of each allocation unit is the allocation page. It does not contain database rows like the other pages, but contains an array that shows how the other 255 pages are used.

The *sysusages* Table

The database storage information is listed in the table *master..sysusages*. Each row in *master..sysusages* represents a space allocation assigned to a database. Thus, each database has one row in *sysusages* for each time **create database** or **alter database** assigns a fragment of disk space to it.

When Adaptive Server is first installed, *sysusages* contains rows for these *dbids*:

- 1, the *master* database
- 2, the temporary database, *tempdb*
- 3, the *model* database
- 4, the *sybsystemprocs* database

If you installed auditing, the *sybsecurity* database will be *dbid* 5.

➤ ***Note***

If your installation is an upgrade from a pre-release 10.0 SQL Server, *sybsystemprocs* and *sybsecurity* may have different database IDs.

As new databases are created or current databases enlarged, new rows are added to *sysusages* to represent new database allocations.

Here is what *sysusages* might look like on an Adaptive Server with the five system databases and two user databases (with *dbid*s 6 and 7). Both user databases were created with the **log on** option. The database with *dbid* 7 has been given additional storage space with two **alter database** commands:

```
select dbid, segmap, lstart, size, vstart
from sysusages
```

```
dbid   segmap      lstart      size            vstart
------ ----------- ----------- -----------     -------
     1           7           0        1536          4
     2           7           0        1024       2564
     3           7           0        1024       1540
     4           7           0        5120   16777216
     5           7           0       10240   33554432
     6           3           0         512    1777216
     6           4         512         512    3554432
     7           3           0        2048   67108864
     7           4        2048        1024   50331648
     7           3        3072         512   67110912
     7           3        3584        1024   67111424

(10 rows affected)
```

The *segmap* Column

The *segmap* column is a bitmask linked to the *segment* column in the user database's *syssegments* table. Since the *logsegment* in each user database is segment 2, and these user databases have their logs on separate devices, *segmap* contains 4 (2^2) for the devices named in the **log on** statement and 3 for the data segment that holds the system segment ($2^0 = 1$) + default segment ($2^1 = 2$).

Some possible values for segments containing data or logs are listed as follows:

Value	Segment
3	Data only (system and default segments)
4	Log only
7	Data and log

Values higher than 7 indicate user-defined segments. The *segmap* column is explained more fully in the segments tutorial section in Chapter 17, "Creating and Using Segments."

The *lstart*, *vstart*, and *size* Columns

- *lstart* column – contains the starting page number in the database of this allocation unit. Each database starts at logical address 0. If additional allocations have been made for a database, as in the case of *dbid* 7, the *lstart* column reflects this.
- *size* column – contains the number of contiguous 2K pages that are assigned to the same database. The ending logical address of

this portion of the database can be determined by adding the values in *lstart* and *size*.

- *vstart* column – contains the address where the piece assigned to this database begins. The upper 4 bits store the virtual device number (*vdevno*), and the lower 4 bits store the virtual block number. (To obtain the virtual device number, divide *sysusages.vstart* or *sysdevices.low* by 16,777,216, which is 2^{24}.) The value in *vstart* identifies which database device contains the page number of the database, because it falls between the values in the *low* and *high* columns of *sysdevices* for the database device in question.

Getting Information About Database Storage

This section explains how to determine which database devices are currently allocated to databases and how much space each database uses.

Database Device Names and Options

To find the names of the database devices on which a particular database resides, use the system procedure **sp_helpdb** with the database name:

```
sp_helpdb pubs2
```

```
name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     2.0 MB     sa           5 Aug 25, 1997   no options set

device_fragments    size          usage            free kbytes
------------------- ------------- ---------------- -----------
pubdev              2.0 MB        data and log             288

device                 segment
---------------------- ----------------------
pubdev                 default
pubdev                 logsegment
pubdev                 system
```

sp_helpdb reports on the size and usage of the devices used by the named database. The status column lists the database options. These options are described in Chapter 16, “Setting Database Options.”

If you are using the named database, **sp_helpdb** also reports on the segments in the database and the devices named by the segments.

See Chapter 17, "Creating and Using Segments," for more information.

When you use **sp_helpdb** without arguments, it reports information about all databases in Adaptive Server:

```
sp_helpdb
```

```
name          db_size  owner dbid created      status
------------- -------- ----- ---- ------------ -------------------
master          3.0 MB sa       1 Jan 01, 1900 no options set
model           2.0 MB sa       3 Jan 01, 1900 no options set
mydata          4.0 MB sa       7 Aug 25, 1997 no options set
pubs2           2.0 MB sa       6 Aug 23, 1997 no options set
sybsecurity    20.0 MB sa       5 Aug 18, 1997 no options set
sybsystemprocs 10.0 MB sa       4 Aug 18, 1997 trunc log on chkpt
tempdb          2.0 MB sa       2 Aug 18, 1997 select into/
                                                bulkcopy/pllsort
```

Checking the Amount of Space Used

The system procedure **sp_spaceused** provides three reports:

- A summary of space used in the database
- A summary of space used by a table and its indexes and *text/image* storage
- A summary of space used by a table, with separate information on indexes and *text/image* storage.

Checking Space Used in a Database

To get a summary of the amount of storage space used by a database, execute the system procedure **sp_spaceused** in the database:

```
sp_spaceused
```

```
database_name                  database_size
------------------------------ -------------
pubs2                          2.0 MB

reserved      data          index_size      unused
------------- ------------- --------------- --------
1720 KB       536 KB        344 KB          840 KB
```

Table 15-2 describes the columns in the report.

Table 15-2: Columns in sp_spaceused output

Column	Description
database_name	The name of the database being examined.
database_size	The amount of space allocated to the database by **create database** or **alter database**.
reserved	Reports the amount of space that has been allocated to all the tables and indexes created in the database. (Space is allocated to database objects inside a database in increments of 1 extent, or 8 pages, at a time.)
data, *index_size*	Shows how much space has been used by data and indexes.
unused	Shows the amount of space that has been reserved but not yet used by existing tables and indexes.

The sum of the values in the *unused*, *index_size*, and *data* columns should be the figure in the *reserved* column. Subtract *reserved* from *database_size* to get the amount of unreserved space. This space is available for new or existing objects that grow beyond the space that has been reserved for them.

By running **sp_spaceused** regularly, you can monitor the amount of database space available. For example, if the *reserved* value is close to the *database_size* value, you are running out of space for new objects. If the *unused* value is also small, you are running out of space for additional data as well.

Checking Summary Information for a Table

You can also use **sp_spaceused** with a table name as its parameter, and get a report on the space used by that table, like this:

```
sp_spaceused titles
name   rowtotal reserved  data    index_size unused
------ -------- --------- ------- ---------- -----
titles 18       48 KB     6 KB    4 KB       38 KB
```

The *rowtotal* column in this report may be different from the results of running **select count(*)** on the table. This is because **sp_spaceused** computes the value with the built-in function **rowcnt.** That function uses values that are stored in the allocation pages. These values are not updated regularly, however, so they can be very different for

tables with a lot of activity. The **update statistics**, **dbcc checktable**, and **dbcc checkdb** commands update the rows-per-page estimate, so *rowtotal* will be most accurate after one of these commands has been run.

It is a good idea to run **sp_spaceused** regularly on *syslogs*, since the transaction log can grow rapidly if there are frequent database modifications. This is particularly a problem if the transaction log is not on a separate device—in which case, it competes with the rest of the database for space.

Checking Information for a Table and Its Indexes

To see information on the space used by individual indexes, enter:

```
sp_spaceused titles, 1
```

```
index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
titleidind           2 KB       32 KB      24 KB
titleind             2 KB       16 KB      14 KB

name       rowtotal   reserved   data   index_size unused
---------- --------  ---------  ------- ---------- ----------
titles          18      46 KB     6 KB      4 KB      36 KB
```

Space taken up by the *text/image* page storage is reported separately from the space used by the table. The object name for text/image storage is always "t" plus the table name:

```
sp_spaceused blurbs,1
```

```
index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
blurbs               0 KB       14 KB      12 KB
tblurbs              14 KB      16 KB      2 KB

name       rowtotal reserved    data    index_size unused
---------- -------- ----------- ------- ---------- ----------
blurbs             6       30 KB    2 KB      14 KB      14 KB
```

Querying System Table for Space Usage Information

You may want to write some of your own queries for additional information about physical storage. For example, to determine the total number of 2K blocks of storage space that exist on Adaptive Server, you can query *sysdevices*:

```
select sum(high - low)
from sysdevices
where status in (2, 3)

-------------------
               7168
```

A 2 in the *status* column represents a physical device; a 3 represents a physical device that is also a default device.

16 Setting Database Options

This chapter describes how to use database options. Topics include:

- What Are Database Options? 16-1
- Using the sp_dboption Procedure 16-1
- Database Option Descriptions 16-2
- Changing Database Options 16-8
- Viewing the Options on a Database 16-9

What Are Database Options?

Database options control many different aspects of database behavior, such as:

- The behavior of transactions
- Defaults for table columns
- Restrictions to user access
- Performance of recovery and **bcp** operations
- Log behavior

The System Administrator and the Database Owner can use database options to configure the settings for an entire database. In this regard, database options differ from **sp_configure** parameters, which affect the entire server, and **set** options, which affect only the current session or stored procedure.

Using the *sp_dboption* Procedure

Use the system procedure **sp_dboption** to change settings for an entire database. The options remain in effect until they are changed. The **sp_dboption** procedure:

- Displays a complete list of the database options when it is used without a parameter
- Changes a database option when used with parameters

You can change options for user databases only. You cannot change options for the *master* database. To change a database option in a user

database (or to display a list of the database options), execute **sp_dboption** while using the *master* database.

The syntax is:

```
sp_dboption [dbname, optname, {true | false}]
```

To make an option or options take effect for every new database, change the option in the *model* database.

Database Option Descriptions

All users with access to the *master* database can execute the **sp_dboption** procedure with no parameters to display a list of the database options. The report from **sp_dboption** looks like this:

```
sp_dboption

Settable database options.
--------------------
abort tran on log full
allow nulls by default
auto identity
dbo use only
ddl in tran
identity in nonunique index
no chkpt on recovery
no free space acctg
read only
select into/bulkcopy/pllsort
single user
trunc log on chkpt
trunc. log on chkpt.
unique auto_identity index
```

For a report on which options have been set in a particular database, execute the system procedure **sp_helpdb** in that database.

The following sections describe each database option in detail.

abort tran on log full

abort tran on log full determines the fate of a transaction that is running when the last-chance threshold is crossed. The default value is **false**, meaning that the transaction is suspended and is awakened only when space has been freed. If you change the setting to **true**, all user queries that need to write to the transaction log are killed until space in the log has been freed.

allow nulls by default

Setting **allow nulls by default** to **true** changes the default null type of a column from **not null** to **null**, in compliance with the SQL standard. The Transact-SQL default value for a column is **not null**, meaning that null values are not allowed in a column unless **null** is specified in the **create table** or **alter table** column definition.

auto identity

While the **auto identity** option is **true**, a 10-digit IDENTITY column is defined in each new table that is created without specifying either a **primary** key, a **unique** constraint, or an IDENTITY column. The column is not visible when you select all columns with the **select *** statement. To retrieve it, you must explicitly mention the column name, *SYB_IDENTITY_COL*, in the select list.

To set the precision of the automatic IDENTITY column, use the **size of auto identity** configuration parameter.

Though you can set **auto identity** to **true** in *tempdb*, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column.

dbo use only

While the **dbo use only** option is set to **true** (on), only the Database Owner can use the database.

ddl in tran

Setting the **ddl in tran** option to **true** allows the following commands to be used inside a user-defined transaction:

- **alter table** (clauses other than **partition** and **unpartition** are allowed)
- **create default**
- **create index**
- **create procedure**
- **create rule**
- **create schema**
- **create table**

- **create trigger**
- **create view**
- **drop default**
- **drop index**
- **drop procedure**
- **drop rule**
- **drop table**
- **drop trigger**
- **drop view**
- **grant**
- **revoke**

Data definition statements lock system tables for the duration of a transaction, which can result in performance problems. Use them only in short transactions.

The following commands cannot be used in a user-defined transaction under any circumstances:

- **alter database**
- **alter table...partition**
- **alter table...unpartition**
- **create database**
- **disk init**
- **dump database**
- **dump transaction**
- **drop database**
- **load transaction**
- **load database**
- **select into**
- **truncate table**
- **update statistics**

identity in nonunique index

The **identity in nonunique index** option automatically includes an IDENTITY column in a table's index keys so that all indexes created on the table are unique. This database option makes logically nonunique indexes internally unique and allows those indexes to be used to process updatable cursors and isolation level 0 reads.

The table must already have an IDENTITY column for the **identity in nonunique index** option to work either from a **create table** statement or from setting the **auto identity** database option to **true** before creating the table.

Use **identity in nonunique index** if you plan to use cursors and isolation level 0 reads on tables that have nonunique indexes. A unique index ensures that the cursor is positioned at the correct row the next time a **fetch** is performed on that cursor.

Do not confuse the **identity in nonunique index** option with **unique auto_identity index**, which is used to add an IDENTITY column with a unique, nonclustered index to new tables.

no chkpt on recovery

The **no chkpt on recovery** option is set to **true** (on) when an up-to-date copy of a database is kept. In these situations, there is a "primary" database and a "secondary" database. Initially, the primary database is dumped and loaded into the secondary database. Then, at intervals, the transaction log of the primary database is dumped and loaded into the secondary database.

If this option is set to **false** (off)—the default—a checkpoint record is added to the database after it is recovered by restarting Adaptive Server. This checkpoint, which ensures that the recovery mechanism is not rerun unnecessarily, changes the sequence number of the database. If the sequence number of the secondary database has been changed, a subsequent dump of the transaction log from the primary database cannot be loaded into it.

Turning this option on for the secondary database causes it not to get a checkpoint from the recovery process so that subsequent transaction log dumps from the primary database can be loaded into it.

no free space acctg

The **no free space acctg** option suppresses free-space accounting and execution of threshold actions for the non-log segments. This speeds recovery time because the free-space counts will not be recomputed for those segments. It disables updating the rows-per-page value stored for each table, so system procedures that estimate space usage may report inaccurate values.

read only

The **read only** option means that users can retrieve data from the database, but cannot modify anything.

select into/bulkcopy/pllsort

The **select into/bulkcopy/pllsort** option must be set to **on** in order to perform operations that do not keep a complete record of the transaction in the log. These actions are:

- Using the **writetext** utility.
- Doing a **select into** a permanent table.
- Doing a "fast" **bulk copy** with **bcp**. Fast **bcp** is used by default on tables that do not have indexes.
- Performing a parallel sort.

Adaptive Server performs minimal logging for these commands, recording only page allocations and deallocations, but not the actual changes made to the data pages.

You do not have to set **select into/bulkcopy/pllsort on** in order to **select into** a temporary table, since *tempdb* is never recovered. The option does not need to be set in order to run **bcp** on a table that has indexes, because inserts are logged.

After you have run a **select into** command or performed a bulk copy in a database, you will not be able to perform a regular transaction log dump. Once you have made minimally logged changes to your database, you must perform a **dump database**, since changes are not recoverable from transaction logs.

Just setting the **select into/bulkcopy/pllsort** option does not block log dumping, but making minimally logged changes to data does block the use of a regular **dump transaction**. However, you can still use **dump transaction...with no_log** and **dump transaction...with truncate_only.**

By default, the **select into/bulkcopy/pllsort** option is turned off in newly created databases. To change the default, turn this option on in the *model* database.

single user

When **single user** is set to **true**, only one user at a time can access the database. You cannot set **single user** to **true** in tempdb.

trunc log on chkpt

When the **trunc log on chkpt** option is **true** (on), the transaction log is truncated (committed transactions are removed) when the **checkpoint** checking process occurs (usually more than once per minute), if 50 or more rows have been written to the log. The log is **not** truncated if less than 50 rows were written to the log, or if the Database Owner runs the **checkpoint** command manually.

It may be useful to turn this option on while doing development work during which backups of the transaction log are not needed. If this option is off (the default), and the transaction log is never dumped, the transaction log continues to grow, and you may run out of space in your database.

When the **trunc log on chkpt** option is on, you cannot dump the transaction log because changes to your data are not recoverable from transaction log dumps. In this situation, issuing the **dump transaction** command produces an error message instructing you to use **dump database** instead.

By default, the **trunc log on chkpt** option is off in newly created databases. To change the default, turn this option on in the *model* database.

◆ ***WARNING!***

If you set trunc log on chkpt on in *model*, and you need to load a set of database and transaction logs into a newly created database, be sure to turn the option off in the new database.

unique auto_identity index

When the **unique auto_identity index** option is set to **true**, it adds an IDENTITY column with a unique, nonclustered index to new tables. By default, the IDENTITY column is a 10-digit numeric datatype, but you can change this default with the **size of auto identity column** configuration parameter.

Though you can set **unique auto_identity index** to **true** in *tempdb*, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column with a unique index.

The **unique auto_identity index** option provides a mechanism for creating tables that have an automatic IDENTITY column with a unique index that can be used with updatable cursors. The unique index on the table ensures that the cursor is positioned at the correct row after a **fetch**. (If you are using isolation level 0 reads and need to make logically nonunique indexes internally unique so that they can process updatable cursors, use the **identity in nonunique index** option.)

In some cases, the **unique auto_identity index** option can avoid the Halloween Problem for the following reasons:

- Users cannot update an IDENTITY column; hence, it cannot be used in the cursor update.
- The IDENTITY column is automatically created with a unique, nonclustered index so that it can be used for the updatable cursor scan.

For more information about the Halloween Problem, IDENTITY columns, and cursors, see the *Transact-SQL User's Guide*.

Do not confuse the **unique auto_identity index** option with the **identity in nonunique index** option, which is used to make all indexes in a table unique by including an IDENTITY column in the table's index keys.

Changing Database Options

Only a System Administrator or the Database Owner can change a user's database options by executing **sp_dboption**. Users aliased to the Database Owner cannot change database options with **sp_dboption**.

You must be using the *master* database to execute **sp_dboption**. Then, for the change to take effect, you must issue the **checkpoint** command while using the database for which the option was changed.

Remember that no *master* database options can be changed.

To use **sp_dboption** to change the *pubs2* database to **read only**:

```
use master
sp_dboption pubs2, "read only", true
```

Then run the **checkpoint** command in the database that was changed:

```
use pubs2
checkpoint
```

For the *optname* parameter of **sp_dboption**, Adaptive Server understands any unique string that is part of the option name. To set the **trunc log on chkpt** option, you can issue this command:

```
use master
sp_dboption pubs2, trunc, true
```

If you enter an ambiguous value for *optname*, an error message is displayed. For example, two of the database options are **dbo use only** and **read only**. Using "only" for the *optname* parameter generates a message because it matches both names. The complete names that match the string supplied are printed out so that you can see how to make the *optname* more specific.

More than one database option at a time can be turned on. You cannot change database options inside a user-defined transaction.

Viewing the Options on a Database

Use the **sp_helpdb** procedure to determine which options are set for a particular database. **sp_helpdb** lists each of the active database options in the "status" column of its output.

The following example shows that the **read only** option is turned on in *mydb*:

```
sp_helpdb mydb
```

```
name   db_size  owner  dbid  created         status
-----  -------  -----  ----  ------------  ----------------------
mydb   2.0 MB    sa     5    Mar 05, 1995  read only

device_fragments      size    usage           free kbytes
-----------------     ------  ------------  -------------
master                2.0 MB  data and log            576

device                         segment
------------------------------ ------------------------------
master                         default
master                         logsegment
master                         system
```

```
name    attribute_class attribute  int_value char_value
        comments
------- --------------- ---------- --------- -------------------------
        -----------
pubs2   buffer manager  cache name      NULL cache for database mydb
        NULL
```

To display a summary of the options for all databases, use **sp_helpdb** without specifying a database:

```
sp_helpdb
```

```
name                 db_size         owner         dbid
   created        status
-------------------  --------------  -------------  -----
   --------------  --------------------------

mydb                   2.0 MB           sa            5
   May 10, 1997   read only

master                 3.0 MB           sa            1
   Jan 01, 1997   no options set

model                  2.0 MB           sa            3
   Jan 01, 1997   no options set

sybsystemprocs         2.0 MB           sa            4
   Mar 31, 1995   trunc log on chkpt

tempdb                 2.0 MB           sa            2
   May 04, 1997   select into/bulkcopy/pllsort
```

17 Creating and Using Segments

This chapter introduces the system procedures and commands for using segments, or named collections of devices, in databases. Topics include:

- What Is a Segment? 17-1
- Commands and Procedures for Managing Segments 17-3
- Why Use Segments? 17-3
- Creating Segments 17-7
- Changing the Scope of Segments 17-8
- Assigning Database Objects to Segments 17-10
- Dropping Segments 17-16
- Getting Information About Segments 17-16
- Segments and System Tables 17-18
- A Segment Tutorial 17-19

See also Chapter 17, "Controlling Physical Data Placement," in the *Performance and Tuning Guide* for information about how segments can improve system performance.

What Is a Segment?

Segments are named subsets of the database devices that are available to a particular Adaptive Server database. A segment can best be described as a label that points to one or more database devices. Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. Using segments can increase Adaptive Server performance and give the System Administrator or Database Owner increased control over the placement, size, and space usage of database objects.

You create segments within a database to describe the database devices that are allocated to the database. Each Adaptive Server database can contain up to 32 segments, including the system-defined segments (see "System-Defined Segments" on page 17-2). Before assigning segment names, you must initialize the database devices with **disk init** and then make them available to the database with **create database** or **alter database**. See Chapter 15, "Creating and

Managing User Databases" for information about **create database** and **alter database.**

System-Defined Segments

When you first create a database, Adaptive Server creates three segments in the database, as described in Table 17-1.

Table 17-1: System-defined segments

Segment	Function
system	Stores the database's system tables.
logsegment	Stores the database's transaction log.
default	Stores all other database objects—unless you create additional segments and store tables or indexes on the new segments by using **create table...on** *segment_name* or **create index...on** *segment_name*.

If you create a database on a single database device, the *system*, *default*, and *logsegment* segments label the same device. If you use the **log on** clause to place the transaction log on a separate device, the segments resemble those shown in Figure 17-1.

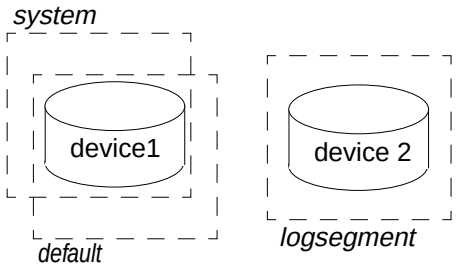


Figure 17-1: System-defined segments

Although you can add and drop user-defined segments, you cannot drop the default, system, or log segments from a database. A database must have at least one default, system-defined, and log segment.

Commands and Procedures for Managing Segments

Table 17-2 summarizes the Adaptive Server commands and system procedures for managing segments.

Table 17-2: Commands and procedures for managing segments

Command or Procedure	Function
sp_addsegment	Defines a segment in a database
create table and **create index**	Creates a database object on a segment
sp_dropsegment	Removes a segment from a database or removes a single device from the scope of a segment
sp_extendsegment	Adds devices to an existing segment
sp_placeobject	Assigns future space allocations for a table or an index to a specific segment
sp_helpsegment	Displays the segment allocation for a database or data on a particular segment
sp_helpdb	Displays the segments on each database device. See Chapter 15, "Creating and Managing User Databases," for examples.
sp_help	Displays information about a table, including the segment on which the table resides
sp_helpindex	Displays information about a table's indexes, including the segments on which the indexes reside

Why Use Segments?

When you add a new device to a database with **alter database,** Adaptive Server places the new device in a default pool of space (the database's *default* and *system* segments). This increases the total space available to the database, but it does not determine which objects will occupy that new space. Any table or index might grow to fill the entire pool of space, leaving critical tables with no room for expansion. It is also possible for several heavily used tables and indexes to be placed on a single physical device in the default pool of space, resulting in poor I/O performance.

Segments provide a way for System Administrators and Database Owners to control the placement of database objects on database devices. When you create an object on a segment, the object can use

all the database devices that are available in the segment, but no other devices. You can use segments to control the space that is available to individual objects. More importantly, you can use it to improve Adaptive Server performance.

The following sections describe how to use segments to control disk space usage and to improve performance. "Moving a Table to Another Device" on page 17-7 explains how to move a table from one device to another using segments and clustered indexes.

Controlling Space Usage

If you assign noncritical objects to a segment, those objects cannot grow beyond the space available in the segment's devices. Conversely, if you assign a critical table to a segment, and the segment's devices are not available to other segments, no other objects will compete with that table for space.

When the devices in a segment become full, you can extend the segment to include additional devices or device fragments as needed. Segments also allow you to use thresholds to warn you when space becomes low on a particular database segment.

Segments and Thresholds

Thresholds monitor the amount of free space in a database segment, and can perform actions automatically when the space becomes full. Each database that stores its transaction log on a separate device from its data has at least one threshold: the last-chance threshold. If you create additional segments for data, you can create new threshold procedures for each segment. See Chapter 23, "Managing Free Space with Thresholds," for more information on thresholds.

Improving Performance

In a large, multidatabase and/or multidrive Adaptive Server environment, careful attention to the allocation of space to databases and the placement of database objects on physical devices can enhance system performance. Ideally, each database has exclusive use of database devices, that is, it does not share a physical disk with another database. In most cases, you can improve performance by placing heavily used database objects on dedicated physical disks or by "splitting" large tables across several physical disks.

The following sections describe these ways to improve performance. See also the *Performance and Tuning Guide* for more information about how segments can improve performance.

Separating Tables, Indexes, and Logs

Generally, placing a table on one physical device, its nonclustered indexes on a second physical device, and the transaction log on a third physical device can speed up performance. Using separate physical devices (disk controllers) reduces the time required to read or write to the disk, since it usually reduces disk head travel. If you cannot devote entire devices in this way, at least restrict all nonclustered indexes to a dedicated physical device.

The **log on** extension to **create database** (or **sp_logdevice**) handles the placement of the transaction log on a separate physical disk. Use segments to place tables and indexes on specific physical devices. See "Assigning Database Objects to Segments" on page 17-10 for information about placing tables and indexes on segments.

Splitting Tables

Splitting a large, heavily used table across devices on separate disk controllers can improve the overall read performance of a table. When a large table exists on multiple devices, it is more likely that small, simultaneous reads will take place on different disks. Figure 17-2 shows a table that is split across the two devices in its segment.

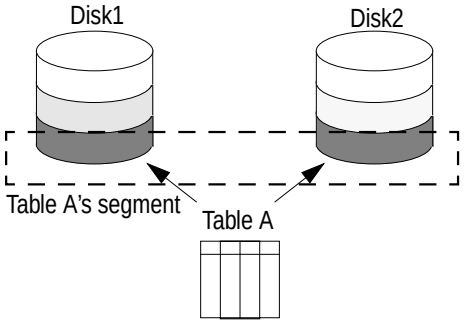


Figure 17-2: Partitioning a table across physical devices

You can split a table across devices using one of three different methods, each of which requires the use of segments:

- Use table partitioning if the table does not have a clustered index.
- Use partial loading if the table has a clustered index.
- Separate the text chain from other data if the table contains *text* or *image* datatypes.

Partitioning Tables

Partitioning a table creates multiple page chains for the table and distributes those page chains over all the devices in the table's segment (see Figure 17-2). Partitioning a table increases both insert performance and read performance, since multiple page chains are available for insertions.

Before you can partition a table, you must create the table on a segment that contains the desired number of devices. The remainder of this chapter describes how to create and modify segments. See "Commands for Partitioning Tables" in Chapter 17, "Controlling Physical Data Placement," of the *Performance and Tuning Guide* for information about partitioning tables using the **alter table** command.

You cannot partition tables that have clustered indexes.

Partial Loading

If you want to split a table that has a clustered index, you can use **sp_placeobject** with multiple **load** commands to load different parts of the table onto different segments. This method can be difficult to execute and maintain, but it provides a way to split tables and their clustered indexes across physical devices. See "Placing Existing Objects on Segments" on page 17-12 for more information and syntax.

Separating text and image Columns

Adaptive Server stores the data for *text* and *image* columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table's other data. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain and base table data on a separate physical device can improve performance. See "Placing Text Pages on a Separate Device" on page 17-15 for more information and syntax.

Moving a Table to Another Device

You can also use segments to move a table from one device to another using the **create clustered index** command. Clustered indexes, where the bottom or **leaf level** of the index contains the actual data, are on the same segment as the table. Therefore, you can completely move a table by dropping its clustered index (if one exists), and creating or re-creating a clustered index on the desired segment. See "Creating Clustered Indexes on Segments" on page 17-15 for more information and syntax.

Creating Segments

Two preliminary steps are needed to create a segment in a database:

- Initialize the physical device with **disk init**.
- Make the database device available to the database by using the **on** clause to **create database** or **alter database**. This automatically adds the new device to the database's *default* and *system* segments, as described under "System-Defined Segments" on page 17-2.

Once the database device exists and is available to the database, define the segment in the database with the stored procedure **sp_addsegment**. The syntax is:

```
sp_addsegment segname, dbname, devname
```

where:

- *segname* is any valid identifier. It is used in **create table** and **create index** statements to create those objects on the segment and on the device named in *devname*. Give segments names that identify what they are used for, and use extensions like "_seg."
- *dbname* is the name of the database where the segment will be created.
- *devname* is the name of the database device—the name used in **disk init** and the **create** and **alter database** statements.

This statement creates the segment *seg_mydisk1* on the database device *mydisk1*:

```
sp_addsegment seg_mydisk1, mydata, mydisk1
```

Changing the Scope of Segments

To perform the tasks described in "Why Use Segments?" on page 17-3, you will also need to manage the scope of segments—the number of database devices to which a segment points. You can:

- Extend the scope of a segment, by making it point to an additional device or devices, or
- Reduce the scope of a segment, by making it point to fewer devices.

Extending the Scope of Segments

You may need to extend a segment if the database object or objects assigned to the segment run out of space. The system procedure **sp_extendsegment** extends the size of a segment by including additional database devices as part of an existing segment. The syntax is:

```
sp_extendsegment segname, dbname, devname
```

Before you can extend a segment:

- The database device must be listed in *sysdevices* (through the use of **disk init**),
- The database device must be available in the desired database (through an **alter database** or **create database** statement), and
- The segment name must exist in the current database (through earlier use of **sp_addsegment**).

The following example adds the database device *pubs_dev2* to an existing segment named *bigseg*:

```
sp_extendsegment bigseg, pubs2, pubs_dev2
```

The word "default" is a keyword, so if you want to extend the *default* segment in your database, you must place the word "default" in quotes:

```
sp_extendsegment "default", mydata, newdevice
```

Automatically Extending the Scope of a Segment

If you use **alter database** to add space on a database device that is new to the database, the *system* and *default* segments are extended to include the new space. Thus, the scope of the *system* and *default*

segments is extended each time you add a new device to the database.

If you use **alter database** to assign additional space on an existing database device, all the segments mapped to the existing device are extended to include the new device fragment. For example, assume that you initialized a 4MB device named *newdev*, allocated 2MB of the device to *mydata*, and assigned the 2MB to the *testseg* segment:

```
alter database mydata on newdev = 2
sp_addsegment testseg, mydata, newdev
```

If you alter *mydata* later to use the remaining space on *newdev*, the remaining space fragment is automatically mapped to the *testseg* segment:

```
alter database mydata on newdev = 2
```

See "A Segment Tutorial" on page 17-19 for more examples about how Adaptive Server assigns new device fragments to segments.

Reducing the Scope of a Segment

You may need to reduce the scope of a segment if it includes database devices that you want to reserve exclusively for other segments. For example, if you add a new database device that is to be used exclusively for one table, you will want to reduce the scope of the *default* and *system* segments so that they no longer point to the new device.

Use the **sp_dropsegment** system procedure to drop a single database device from a segment, reducing the segment's scope. The following syntax removes a single database device from a segment:

```
sp_dropsegment segname, dbname, device
```

With three arguments, **sp_dropsegment** does not drop the segment, but drops only the given *device* from the scope of devices spanned by the segment. You can also use **sp_dropsegment** to remove an entire segment from the database, as described under "Dropping Segments" on page 17-16.

The following example removes the database device *pubs_dev2* from the scope of *bigseg*:

```
sp_dropsegment bigseg, pubs2, pubs_dev2
```

Assigning Database Objects to Segments

This section explains how to assign new or existing database objects to user-defined segments in order to:

- Restrict new objects to one or more database devices
- Place a table and its index on separate devices to improve performance
- Split an existing object over multiple database devices

This section uses the techniques described in "Creating Segments" and "Changing the Scope of Segments" on page 17-8.

Creating New Objects on Segments

To place a new object on a segment, first follow the instructions for "Creating Segments" on page 17-7 to create the new segment. You may also want to change the scope of this segment (or other segments) so that it points only to the desired database devices. Remember that when you add a new database device to a database, it is automatically added to the scope of the *default* and *system* segments.

After you have defined the segment in the current database, use the **create table** or **create index** command with the optional **on** *segment_name* clause to create the object on the segment. The syntax is:

```
create table table_name (col_name datatype ... )
   [on segment_name]

create [ clustered | nonclustered ] index index_name
   on table_name(col_name)
   [on segment_name]
```

➤ *Note*

Clustered indexes, where the bottom leaf, or leaf level, of the index contains the actual data, are by definition on the same segment as the table. See "Creating Clustered Indexes on Segments" on page 17-15.

Example: Creating a table and index on separate segments

Figure 17-3 on page 17-11 summarizes the sequence of Transact-SQL commands used to create tables and indexes on specific physical disks:

1. Start by using the *master* database.
2. Initialize the physical disks.
3. Allocate the new database devices to a database.
4. Change to the *mydata* database by using the **use** *database* command.
5. Create two new segments, each of which points to one of the new devices.
6. Reduce the scope of the *default* and *system* segments so that they do not point to the new devices.
7. Create the objects, giving the new segment names.

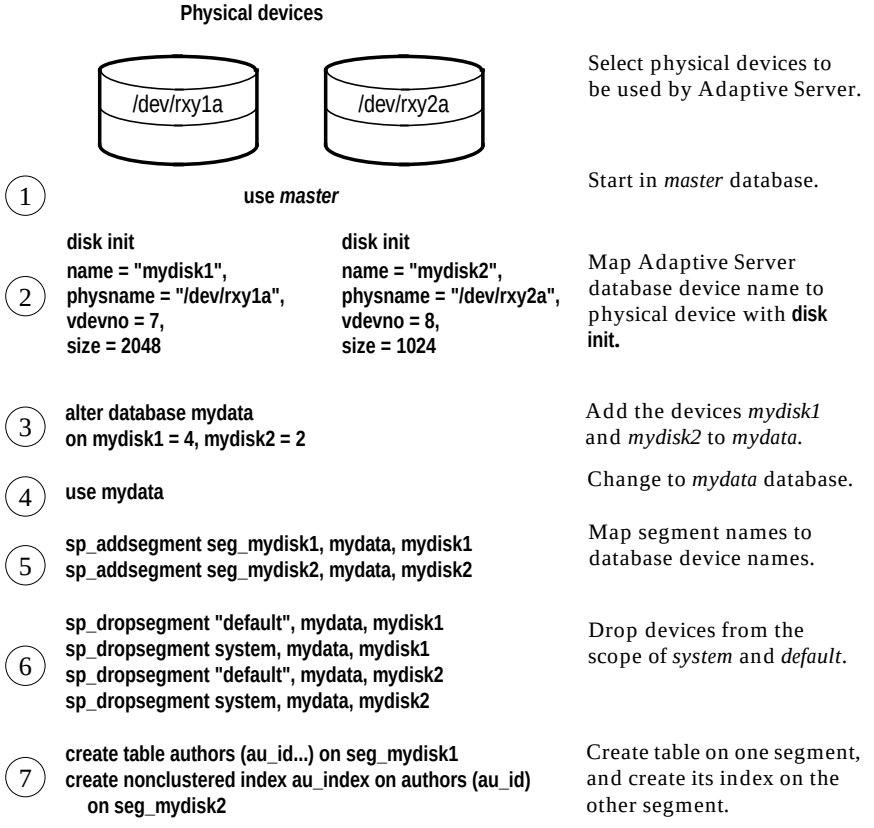


Figure 17-3: Creating objects on specific devices using segments

Placing Existing Objects on Segments

The system procedure **sp_placeobject** does not remove an object from its allocated segment. However, it causes all further disk allocation for that object to occur on the new segment it specifies. The syntax is:

```
sp_placeobject segname, objname
```

The following command causes all further disk allocation for the *mytab* table to take place on *bigseg*:

```
sp_placeobject bigseg, mytab
```

➤ ***Note***

sp_placeobject does not move an object from one database device to another. Whatever pages have been allocated on the first device remain allocated; whatever data was written to the first device remains on the device. **sp_placeobject** affects only future space allocations.

To completely move a table, you can drop its clustered index (if one exists), and create or re-create a clustered index on the desired segment. To completely move a nonclustered index, drop the index and re-create it on the new segment. See “Creating Clustered Indexes on Segments” on page 17-15 for instructions on moving a table.

After you have used **sp_placeobject**, executing **dbcc checkalloc** causes the following message to appear for each object that is split across segments:

```
Extent not within segment: Object object_name,
indid index_id includes extents on allocation page
page_number which is not in segment segment_name.
```

You can ignore this message.

Example: Splitting a table and its clustered index across physical devices

Performance can be improved for high-volume, multiuser applications when large tables are split across segments that are located on separate disk controllers.

Figure 17-4 on page 17-14 summarizes the process of splitting a table across two segments, as shown in the following steps:

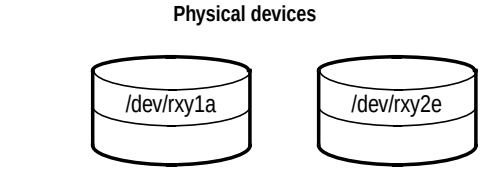
1. Begin by using the *master* database.
2. Initialize the devices with **disk init**.

3. Assign both devices to the *mydata* database with **alter database**.
4. Change to the *mydata* database by entering the **use** *database* command.
5. Create three segments. The first two should each point to one of the new devices. Extend the scope of the third segment so that it labels both devices.
6. Drop the *system* and *default* segments from both devices.
7. Create the table and its clustered index on the first segment.
8. Load half of the table's data onto the first segment.
9. Use **sp_placeobject** to cause all further allocations of disk space to occur on the second segment.
10. Load the remaining data onto the second segment.
11. Once the data has been loaded onto the second segment, use **sp_placeobject** again to place the table on the segment that spans both devices.

➤ ***Note***

The order of steps is quite important at certain stages. In particular, the clustered index **must** be created before the table is placed on the second segment. After that, creating a clustered index would cause the entire object to migrate to the second segment. If the index is created after the table is placed on *seg_bothdisks*, the allocation of disk space is unpredictable.

In the preceding example, the balance of disk allocation may change over time if the table is updated frequently. To guarantee that the speed advantages are maintained, it may be necessary to drop and re-create the table at some point.



Select physical devices to be used by Adaptive Server.

1. **use master**

   Start in *master* database.

2. **disk init**
   name = "mydisk1",
   physname = "/dev/rxy1a",
   vdevno = 7,
   size = 2048

   disk init
   name = "mydisk2",
   physname = "/dev/rxy2e",
   vdevno = 8,
   size = 2048

   Map Adaptive Server database device name to physical device with **disk init**.

3. **alter database mydata**
   on mydisk1 = 4, mydisk2 = 4

   Add the devices *mydisk1* and *mydisk2* to *mydata*.

4. **use mydata**

   Change to *mydata* database.

5. **sp_addsegment seg_mydisk1, mydata, mydisk1**
   sp_addsegment seg_mydisk2, mydata, mydisk2
   sp_addsegment seg_bothdisks, mydata, mydisk1
   sp_extendsegment seg_bothdisks, mydata, mydisk2

   Add a segment on *mydisk1* and another on *mydisk2*. Create a third segment, and extend it to span both disks.

6. **sp_dropsegment "default", mydata, mydisk1**
   sp_dropsegment system, mydata, mydisk1
   sp_dropsegment "default", mydata, mydisk2
   sp_dropsegment system, mydata, mydisk2

   Drop devices from the scope of *system* and *default*.

7. **create table authors (au_id etc.) on seg_mydisk1**
   create clustered index au_ind on authors (au_id)
   on seg_mydisk1

   Create the table and clustered index on the segment.

8. [use **bcp** to load half of the rows]

   Load half of the rows.

9. **sp_placeobject segmydisk2, authors**

   Place the object on the second segment.

10. [use **bcp** to load the rest of the rows]

    Load the rest of the rows.

11. **sp_placeobject seg_bothdisks, authors**

    Place the table on the segment that spans both disks.

Figure 17-4: Splitting a large table across two segments

Placing Text Pages on a Separate Device

When you create a table with *text* or *image* columns, the data is stored on a separate chain of text pages. A table that contains *text* or *image* columns has an additional entry in *sysindexes* for the text chain, with the name column set to the name of the table preceded by the letter "t" and an *indid* of 255. You can use **sp_placeobject** to store the text chain on a separate device, giving both the table name and the name of the text chain from *sysindexes*:

```
sp_placeobject textseg, "mytab.tmytab"
```

➤ ***Note***

By default, a chain of text pages is placed on the same segment as its table. After you execute **sp_placeobject**, pages that were previously written on the old device remain allocated, but all new allocations take place on the new segment.

Creating Clustered Indexes on Segments

The bottom, or leaf level, of a clustered index contains the data. Therefore, a table and its clustered index are on the same segment. If you create a table on one segment and create its clustered index on a different segment, the table travels with its index. The entire table will leave the segment on which the table was created and migrate to the segment where the clustered index was created. This provides a quick and easy way to move a table to other devices in your database.

The syntax for creating a clustered index on a segment is:

```
create [unique] clustered index index_name
   on [[database.]owner.]table_name (column_name
        [, column_name]...)
   [with {fillfactor = x, ignore_dup_key, sorted_data,
     [ignore_dup_row | allow_dup_row]}]
   on segment_name
```

See "Segments and Clustered Indexes" on page 17-24 for an example of this command.

Dropping Segments

When you use the stored procedure **sp_dropsegment** with only a segment name and the database name, the named segment is dropped from the database. However, you cannot drop a segment as long as database objects are still assigned to it. You must assign the objects to another segment (as described under "Assigning Database Objects to Segments" on page 17-10) or drop the objects first and then drop the segment. Use the commands described under "Getting Information About Segments" on page 17-16 to determine which objects are still assigned to a segment.

The syntax for dropping a segment is:

```
sp_dropsegment segname, dbname
```

Moreover, you cannot completely drop the default, system, or log segment from a database. A database must have at least one default, system, and log segment. You can reduce the scope of these segments only (by using **sp_dropsegment** with three arguments). See "Reducing the Scope of a Segment" on page 17-9 for details about reducing the scope of a segment.

➤ ***Note***

Dropping a segment removes its name from the list of segments in the database, but it does not remove database devices from the allocation for that database, nor does it remove objects from devices.

If you drop all segments from a database device, the space is still allocated to the database but cannot be used for database objects. **dbcc checkcatalog** reports "Missing segment in Sysusages segmap." To make a device available to a database, use **sp_extendsegment** to map the device to the database's default segment:

sp_extendsegment "default", *dbname, devname*

Getting Information About Segments

Four system procedures provide information about segments:

- **sp_helpsegment** lists the segments in the database in which it is executed or displays information about a particular segment in the database.
- **sp_helpdb** displays information about the relationship between devices and segments in a database.

- **sp_help** and **sp_helpindex** display information about tables and indexes, including the segment to which the object is assigned.

sp_helpsegment

The system procedure **sp_helpsegment**, when used without an argument, displays information about all of the segments in the database where you execute it:

```
sp_helpsegment
```

```
segment name                          status
------- ----------------------------- ------
      0 system                             0
      1 default                            1
      2 logsegment                         0
      3 seg1                               0
      4 seg2                               0
```

For information about a particular segment, specify the segment name as an argument. Use quotes when requesting information about the *default* segment:

```
sp_helpsegment "default"
```

The following example displays information about *seg1*:

```
sp_helpsegment seg1
```

```
segment name                          status
------- ----------------------------- ------
      4 seg1                               0

device                 size             free_pages
---------------------- ---------------- -----------
user_data10            15.0MB                 6440
user_data11            15.0MB                 6440
user_data12            15.0MB                 6440

table_name             index_name           indid
---------------------- -------------------- ------
customer               customer                 0

total_size      total_pages free_pages  used_pages
--------------- ----------- ----------- -----------
45.0MB                23040       19320        3720
```

sp_helpdb

When you execute **sp_helpdb** within a database, and give that database's name, the system procedure displays information about the segments in the database.

For example:

```
sp_helpdb mydata
```

```
name      db_size    owner  dbid created        status
--------- ---------- ----------- -------------- ------------
mydata        8.0 MB sa        4 May 27, 1993   no options set

device_fragments    size       usage             free kbytes
------------------- ---------- ----------------- -----------
datadev2            4.0 MB     data only                3408
logdev              2.0 MB     log only                 2032
seg_mydisk1         2.0 MB     data only                2016

device                         segment
------------------------------ -------------------------
datadev2                       default
datadev2                       system
logdev                         logsegment
seg_mydisk1                    seg1
```

sp_help and *sp_helpindex*

When you execute **sp_help** and **sp_helpindex** in a database, and give a table's name, the system procedure displays information about which segment(s) stores the table or its indexes.

For example:

```
sp_helpindex authors
```

```
index_name     index_description                    index_keys
------------   ---------------------------------    ----------
au_index       nonclustered located on seg_mydisk2  au_id
```

Segments and System Tables

Three system tables store information about segments: the *master..sysusages* table and two system tables in the user database, *sysindexes* and *syssegments*. The system procedure **sp_helpsegment** uses these three tables to provide its information. In addition, it finds the database device name in *sysdevices*.

When you allocate a device to a database with **create database** or **alter database**, Adaptive Server adds a row to *master..sysusages*. The *segmap* column in *sysusages* provides bitmaps to the segments in the database for each device.

create database also creates the *syssegments* table in the user database with these default entries:

```
segment name            status
------- --------------- ------
      0 system              0
      1 default             1
      2 logsegment          0
```

When you add a segment to a database with **sp_addsegment**, the procedure:

- Adds a new row to the *syssegments* table in the user database, and
- Updates the *segmap* in *master..sysusages*.

When you create a table or an index, Adaptive Server adds a new row to *sysindexes*. The *segment* column in that table stores the segment number, showing where the server will allocate new space for the object. If you do not specify a segment name when you create the object, it is placed on the *default* segment; otherwise, it is placed on the specified segment.

If you create a table containing *text* or *image* columns, a second row is also added to *sysindexes* for the linked list of text pages; by default, the chain of text pages is stored on the same segment as the table. An example using **sp_placeobject** to put the text chain on its own segment is included under "A Segment Tutorial" on page 17-19.

The *name* from *syssegments* is used in **create table** and **create index** statements. The *status* column indicates which segment is the default segment.

➤ ***Note***

See "System Tables That Manage Space Allocation" on page 15-15 for more information about the *segmap* column and the system tables that manage storage.

A Segment Tutorial

Segments provide a flexible tool for allowing System Administrators to assign objects to particular database devices. For example, you can

add a database device to a database, and improve system performance, by assigning a high-use table to the device. To achieve these improvements, you must be certain that no other database objects use the new segment. The following tutorial shows how to create a user segment and how to remove all other segment mappings from the device.

Remember the following when you are working with segments and devices:

- If you assign the space in fragments, for example, some of it with **create database** and some with **alter database**, each fragment will have an entry in *sysusages*.
- When an additional fragment of a device is assigned to a database that has already been assigned a fragment of that device, all segments mapped to the existing fragment are mapped to the new fragment.
- If you use **alter database** to add space on a device that is new to the database, the *system* and *default* segments are automatically mapped to the new space.

The tutorial begins with a new database, created with one device for the database objects and another for the transaction log:

```
create database mydata on bigdevice = 4
     log on logdev = 2
```

Now, if you **use mydata**, and run **sp_helpdb**, you will see:

```
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       6.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage           free kbytes
---------------------- ------------- --------------- -----------
bigdevice              4.0 MB        data only              3408
logdev                 2.0 MB        log only               2032

device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
```

Like all newly created databases, *mydata* has the segments named *default*, *system*, and *logsegment*. Since the **create database** statement used

log on, the *logsegment* is mapped to its own device, *logdev*, and the *default* and *system* segments are both mapped to *bigdevice*.

If you add space on the same database devices to *mydata*, and then run **sp_helpdb** again, you will see entries for the added fragments:

```
use master

alter database mydata on bigdevice = 2
    log on logdev = 1

use mydata

sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       9.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage           free kbytes
---------------------- ------------- --------------- -----------
bigdevice              2.0 MB        data only              2048
bigdevice              4.0 MB        data only              3408
logdev                 1.0 MB        log only               1024
logdev                 2.0 MB        log only               2032

device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
```

Always add log space to log space and data space to data space. Adaptive Server will instruct you to use **with override** if you try to allocate a segment that is already in use for data to the log, or vice versa. Remember that segments are mapped to entire devices, and not just to the space fragments. If you change any of the segment assignments on a device, you make the change for all of the fragments.

When you allocate a new device to the database with **alter database**—one that is not in use by the database—the new fragments are automatically assigned:

- To the *system* and *default* segments, if they are in the **on** clause or if they are default devices, or
- To the log segment, if they are in the **log on** clause.

The following example allocates a new database device that has not been used by *mydata*:

```
use master
```

```
alter database mydata on newdevice = 3
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata      12.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage           free kbytes
---------------------- ------------- --------------- -----------
bigdevice              2.0 MB        data only              2048
bigdevice              4.0 MB        data only              3408
logdev                 1.0 MB        log only               1024
logdev                 2.0 MB        log only               2032
newdevice              3.0 MB        data only              3072

device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
newdevice              default
newdevice              system
```

The *default* and *system* segments are automatically mapped to the new space. In some cases, you want the new space for use as default storage space for **create table** or **create index** statements. In that case, this allocation is fine. However, if you are adding space in order to assign a table or index to a specific segment (and, therefore, to specific devices), you will want to reduce the scope of the *system* and *default* segments to drop the new fragment and add your own segment name.

The following example creates a segment called *new_space* on *newdevice*:

```
sp_addsegment new_space, mydata, newdevice
```

Here is just the portion of the **sp_helpdb** report that has changed, the portion that lists the segment mapping:

```
device                         segment
------------------------------ ------------------
bigdevice                      default
bigdevice                      system
logdev                         logsegment
newdevice                      default
newdevice                      new_space
newdevice                      system
```

Notice that the *default* and *system* segments are still mapped to *newdevice*. If you are planning to use *new_space* to store a user table or index for improved performance, and you want to ensure that other user objects are not stored on the device by default, reduce the scope of the *default* and *system* segments with **sp_dropsegment**:

```
sp_dropsegment system, mydata, newdevice

sp_dropsegment "default", mydata, newdevice
```

The quotes around "default" are needed because it is a Transact-SQL reserved word.

Here is just the portion of the **sp_helpdb** report that shows the segment mapping:

```
device                          segment
------------------------------ --------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       new_space
```

Only *new_space* is now mapped to *newdevice*. Users who create objects can use **on** *new_space* to place a table or index on the device that corresponds to that segment. Since the *default* segment is not pointing to that database device, users who create tables and indexes without using the **on** clause will not be placing them on your specially prepared device.

If you use **alter database** on *newdevice* again, the new space fragment acquires the same segment mapping as the existing fragment of that device (that is, the *new_space* segment only).

At this point, if you use **create table** and name *new_space* as the segment, you will get results like these from **sp_help** and **sp_helpsegment**:

```
create table mytabl (c1 int, c2 datetime)
    on new_space

sp_help mytabl
```

```
Name              Owner             Type
----------------- ----------------- ----------------
mytabl            dbo               user table

Data_located_on_segment        When_created
------------------------------ --------------------
new_space                      May 27 1993  3:21PM

Column_name   Type      Length Nulls Default_name Rule_name
------------- --------- ------ ----- ------------ ----------
c1            int            4     0 NULL           NULL
c2            datetime       8     0 NULL           NULL
Object does not have any indexes.
No defined keys for this object.
```

```
sp_helpsegment new_space
segment name                          status
------- ------------------------------ ------
      3 new_space                           0

device                 size           free_pages
---------------------- -------------- -----------
newdevice              3.0MB                1528

table_name            index_name             indid
--------------------- ---------------------- ------
mytabl                mytabl                      0

total_size       total_pages free_pages  used_pages
---------------- ----------- ----------- -----------
3.0MB                  1536        1528           8
```

Segments and Clustered Indexes

As mentioned in "Creating Clustered Indexes on Segments" on page 17-15, if you create a table on one segment and create its clustered index on a different segment, the table travels with its index. The following example creates a clustered index, without specifying the segment name, using the same table you just created on the *new_space* segment in the preceding example. Checking *new_space* after the **create index** command shows that no objects remain on the segment:

```
/* Don't try this at home */
create clustered index mytabl_cix
    on mytabl(c1)
```

```
sp_helpsegment new_space
segment name                          status
------- ------------------------------ ------
      3 new_space                           0

device                 size           free_pages
---------------------- -------------- -----------
newdevice              3.0MB                 1528

total_size       total_pages free_pages  used_pages
---------------- ----------- ----------- -----------
3.0MB                    1536        1528           8
```

If you have placed a table on a segment, and you need to create a clustered index, be sure to use the **on** *segment_name* clause, or the table will migrate to the *default* segment.

18 Checking Database Consistency

This chapter describes how to check database consistency and perform some kinds of database maintenance using the **dbcc** commands. Topics include:

- What Is the Database Consistency Checker? 18-1
- Understanding Page and Object Allocation Concepts 18-2
- What Checks Can Be Performed with dbcc? 18-6
- Checking Consistency of Databases and Tables 18-7
- Checking Page Allocation 18-13
- Correcting Allocation Errors Using the fix | nofix Option 18-16
- Checking Consistency of System Tables 18-18
- Strategies for Using Consistency Checking Commands 18-18
- Dropping a Damaged Database 18-26
- Preparing to Use dbcc checkstorage 18-27
- Consistency Checking with dbcc checkstorage 18-10
- Maintaining dbccdb 18-38
- Generating Reports from dbccdb 18-41

What Is the Database Consistency Checker?

The Database Consistency Checker (**dbcc**) provides commands for checking the logical and physical consistency of a database. Two major functions of **dbcc** are:

- Checking page linkage and data pointers at both the page level and the row level using **checkstorage** or **checktable** and **checkdb**
- Checking page allocation using **checkstorage** or **checkalloc**, **tablealloc**, and **indexalloc**

dbcc checkstorage stores the results of checks in the *dbccdb* database. You can print reports from *dbccdb* using the **dbcc** stored procedures.

Use the **dbcc** commands:

- As part of regular database maintenance – The integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency

checks on a regular basis. These checks can detect errors, and often correct them before they affect a user's ability to use Adaptive Server.

- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database for additional confidence in the integrity of the backup.
- Because you suspect that a database is damaged – For example, if using a particular table generates the message "Table corrupt," you can use **dbcc** to determine if other tables in the database are also damaged.

If you are using Component Integration Services, there are additional **dbcc** commands you can use for remote databases. See the *Component Integration Services User's Guide* for more information.

Understanding Page and Object Allocation Concepts

When you initialize a database device, the **disk init** command divides the new space into **allocation units** of 256 2K data pages. The first page of each allocation unit is an **allocation page**, which tracks the use of all pages in the allocation unit. Allocation pages have an object ID of 99; they are not real database objects and do not appear in system tables, but **dbcc** errors on allocation pages report this value.

When a table or an index requires space, Adaptive Server allocates a block of 8 2K pages to the object. This 8-page block is called an **extent**. Each 256-page allocation unit contains 32 extents. Adaptive Server uses extents as a unit of space management to allocate and deallocate space as follows:

- When you create a table or an index, Adaptive Server allocates an extent for the object.
- When you add rows to an existing table, and the existing pages are full, Adaptive Server allocates another page. If all pages in an extent are full, Adaptive Server allocates an additional extent.
- When you drop a table or an index, Adaptive Server deallocates the extents it occupied.
- When you delete rows from a table so that it shrinks by a page, Adaptive Server deallocates the page. If the table shrinks off the extent, Adaptive Server deallocates the extent.

Every time space is allocated or deallocated on an extent, Adaptive Server records the event on the allocation page that tracks the extents for that object. This provides a fast method for tracking space allocations in the database, since objects can shrink or grow without excess overhead.

Figure 18-1 shows how data pages are set up within extents and allocation units in Adaptive Server databases.

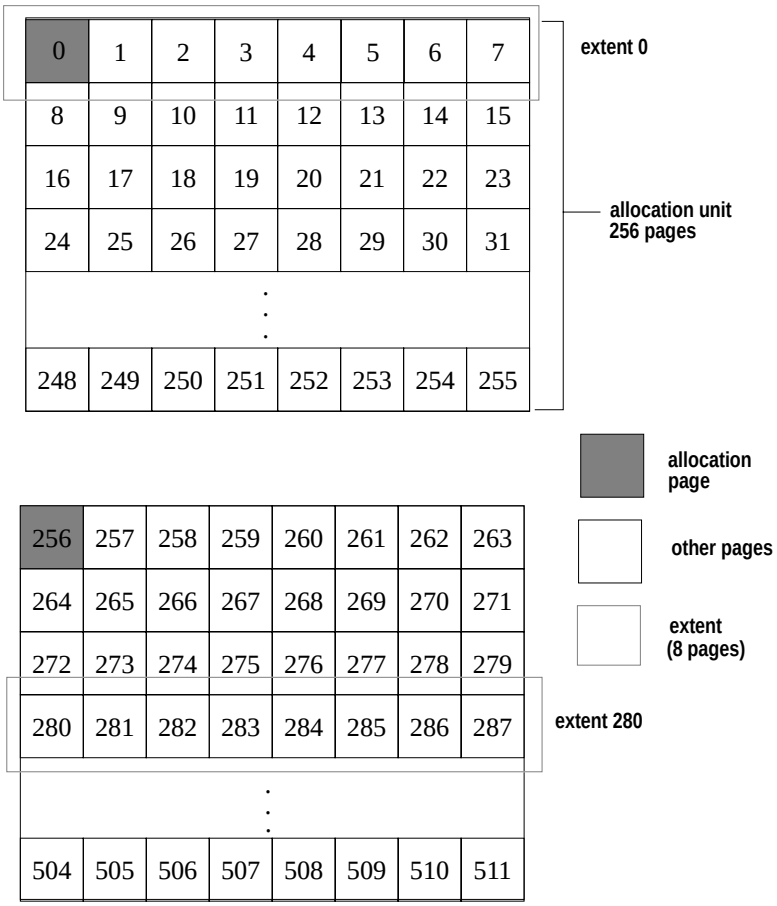


Figure 18-1: Page management with extents

dbcc checkalloc checks all allocation pages (page 0 and all pages divisible by 256) in a database and reports on the allocation information it finds there. **dbcc indexalloc** and **dbcc tablealloc** check allocation for specific database objects.

Understanding the Object Allocation Map (OAM)

Each table and index on a table has an **Object Allocation Map** (**OAM**). The OAM is stored on pages allocated to the table or index and is checked when a new page is needed for the index or table. A single OAM page can hold allocation mapping for between 2,000 and 63,750 data or index pages.

The OAM pages point to the allocation page for each allocation unit where the object uses space. The allocation pages, in turn, track the information about extent and page usage within the allocation unit. In other words, if the *titles* table is stored on extents 24 and 272, the OAM page for the *titles* table points to pages 0 and 256.

Figure 18-2 shows an object stored on 4 extents, numbered 0, 24, 272 and 504. The OAM is stored on the first page of the first segment. In this case, since the allocation page occupies page 0, the OAM is located on page 1.

This OAM points to two allocation pages: page 0 and page 256.

These allocation pages track the pages used in each extent used by all objects with storage space in the allocation unit. For the object in this example, it tracks the allocation and deallocation of pages on extents 0, 24, 272, and 504.

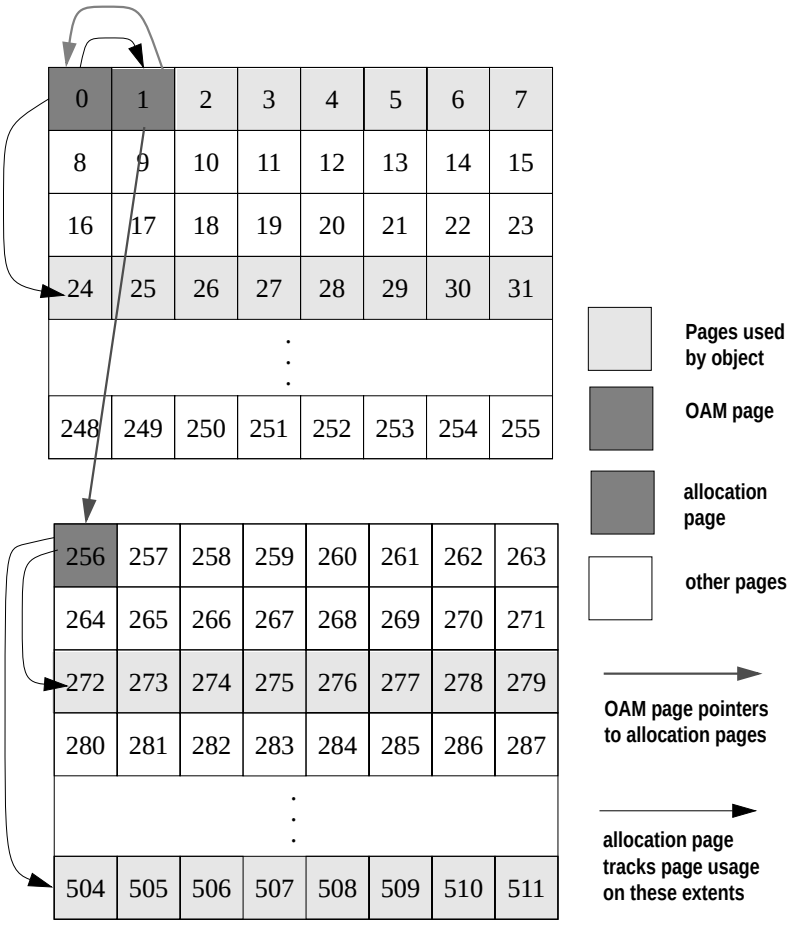


Figure 18-2: OAM page and allocation page pointers

The **dbcc checkalloc** and **dbcc tablealloc** commands examine this OAM page information, in addition to checking page linkage, as described in the section "Understanding Page Linkage" on page 18-5.

Understanding Page Linkage

After a page has been allocated to a table or an index, that page is linked with other pages used for the same object. Figure 18-3

illustrates this linking. Each page contains a header that includes the number of the page that precedes it ("prev") and of the page that follows it ("next"). When a new page is allocated, the header information on the surrounding pages changes to point to that page. **dbcc checktable** and **dbcc checkdb** check page linkage. **dbcc checkalloc**, **tablealloc**, and **indexalloc** compare page linkage to information on the allocation page.

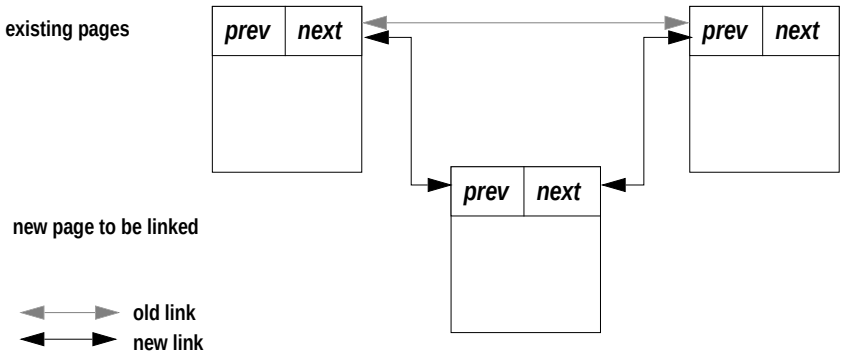


Figure 18-3: How a newly allocated page is linked with other pages

What Checks Can Be Performed with *dbcc*?

Table 18-1 summarizes the checks performed by the **dbcc** consistency checking commands. Table 18-2 on page 18-19 compares the performance and impact of using the different **dbcc** commands.

Table 18-1: Comparison of checks performed by *dbcc* commands

Checks Performed	*check-storage*	*check-table*	*check-db*	*check-alloc*	*index-alloc*	*table-alloc*	*check-catalog*
Checks allocation of text valued columns	X						
Checks index consistency		X	X				
Checks index sort order		X	X				
Checks OAM page entries	X	X	X		X	X	
Checks page allocation	X			X	X	X	
Checks page consistency	X	X	X				

Table 18-1: Comparison of checks performed by *dbcc* commands (continued)

Checks Performed	*check-storage*	*check-table*	*check-db*	*check-alloc*	*index-alloc*	*table-alloc*	*check-catalog*
Checks pointer consistency	X	X	X				
Checks system tables							X
Checks text column chains	X	X	X	X			
Checks text valued columns	X	X	X				

➤ ***Note***

All **dbcc** commands except **dbrepair** and **checkdb** with the **fix** option can be run while the database is active.

Only the table owner can execute **dbcc** with the **checktable**, **fix_text**, or **reindex** keywords. Only the Database Owner can use the **checkstorage**, **checkdb**, **checkcatalog**, **checkalloc**, **indexalloc**, and **tablealloc** keywords. Only a System Administrator can use the **dbrepair** keyword.

Checking Consistency of Databases and Tables

The **dbcc** commands for checking the consistency of databases and tables are:

- **dbcc checkstorage**
- **dbcc checktable**
- **dbcc checkdb**

dbcc checkstorage

Use **dbcc checkstorage** to perform the following checks:

- Allocation of text valued columns
- Page allocation and consistency
- OAM page entries
- Pointer consistency
- Text valued columns and text column chains

The syntax for **dbcc checkstorage** is:

```
dbcc checkstorage [(dbname)]
```

where *dbname* is the name of the **target database** (the database to be checked).

Advantages of Using *dbcc checkstorage*

The **dbcc checkstorage** command:

- Combines many of the checks provided by the other **dbcc** commands
- Does not lock tables or pages for extended periods, which allows **dbcc** to locate errors accurately while allowing concurrent update activity
- Scales linearly with the aggregate I/O throughput
- Separates the functions of checking and reporting, which allows custom evaluation and report generation
- Provides a detailed description of space usage in the target database
- Records **dbcc checkstorage** activity and results in the *dbccdb* database, which allows trend analysis and provides a source of accurate diagnostic information

Comparison of *dbcc checkstorage* and Other *dbcc* Commands

dbcc checkstorage is different from the other **dbcc** commands in that it requires:

- The *dbccdb* database to store configuration information and the results of checks made on the target database. For information on creating *dbccdb*, see "Preparing to Use dbcc checkstorage" on page 18-27.
- At least two workspaces to use during the check operation. For information on the workspaces, see "dbccdb Workspaces" on page 19-13.
- System and stored procedures to help you prepare your system to use **dbcc checkstorage** and to generate reports on the data stored in *dbccdb*. For information on the system and stored procedures, see "Preparing to Use dbcc checkstorage" on page 18-27, "Maintaining dbccdb" on page 18-38 and "Generating Reports from dbccdb" on page 18-41.

The **dbcc checkstorage** command does not replace the other **dbcc** commands, but it combines some of the check functionality.

dbcc checkstorage does not repair any faults. After you run **dbcc checkstorage** and generate a report to see the faults, you can run the appropriate **dbcc** command to repair the faults. Choose the command you want to run based on the kinds of faults reported.

Understanding the *dbcc checkstorage* Operation

The **dbcc checkstorage** operation consists of the following steps:

1. Inspection – **dbcc checkstorage** uses the device allocation and the segment definition of the database being checked to determine the level of parallel processing that can be used. **dbcc checkstorage** also uses the configuration parameters **max worker processes** and **dbcc named cache** to limit the level of parallel processing that can be used.
2. Planning – **dbcc checkstorage** generates a plan for executing the operation that takes advantage of the parallelism discovered in step 1.
3. Execution and optimization – **dbcc checkstorage** uses Adaptive Server worker processes to perform parallel checking and storage analysis of the **target database**. It attempts to equalize the work performed by each worker process and consolidates the work of underutilized worker processes. As the check operation proceeds, **dbcc checkstorage** extends and adjusts the plan generated in step 2 to take advantage of the additional information gathered during the check operation.
4. Reporting and control – During the check operation, **dbcc checkstorage** records in the *dbccdb* database all the faults it finds in the target database for later reporting and evaluation. It also records the results of its storage analysis in *dbccdb*. When **dbcc checkstorage** encounters a fault, it attempts to recover and continue the operation, but ends operations that cannot succeed after the fault. For example, a defective disk does not cause **dbcc checkstorage** to fail; however, check operations performed on the defective disk cannot succeed, so they are not performed.

   dbcc checkstorage might fail in the initialization phase if another session performs a **drop table** command concurrently. If this happens, run **dbcc checkstorage** again when the drop table process is finished.

Performance and Scalability

dbcc checkstorage scales linearly with aggregate I/O throughput for a substantial performance improvement over **dbcc checkalloc**. The scaling property of **dbcc checkstorage** means that if the database doubles in size and the hardware doubles in capacity (realizable I/O throughput), the time required for a **dbcc** check remains unchanged. Doubling the capacity would typically mean doubling the number of disk spindles and providing sufficient additional I/O channel capacity, system bus capacity, and CPU capacity to realize the additional aggregate disk throughput.

Most of the checks performed by using **dbcc checkalloc** and **dbcc checkdb**, including text column chain verification, are achieved with a single check when you use **dbcc checkstorage**, thereby eliminating redundant check operations.

dbcc checkstorage checks the entire database, including unused pages, so execution time is relative to database size. Therefore, when you use **dbcc checkstorage**, there is not a large difference between checking a database that is nearly empty and checking one that is nearly full, as there is with the other **dbcc** commands.

Unlike the other **dbcc** commands, the performance of **dbcc checkstorage** does not depend heavily on data placement. Therefore, the performance of **dbcc checkstorage** is consistent for each session, even if the data placement changes between sessions.

Because **dbcc checkstorage** does extra work to set up the parallel operation and records large amounts of data in *dbccdb*, the other **dbcc** commands will be faster when the target database is small.

The location and allocation of the workspaces used by **dbcc checkstorage** can affect performance and scalability. For more information on how to set up the workspaces to maximize the performance and scalability of your system, see “dbccdb Workspaces” on page 19-13.

Consistency Checking with *dbcc checkstorage*

Use the **dbcc checkstorage** command to start a parallel consistency check on a specific database. For a list of the kinds of checks it performs, see Table 18-1 on page 18-6. For details on how the **dbcc checkstorage** operation works, see “Understanding the dbcc checkstorage Operation” on page 18-9. For information on the syntax, see “dbcc checkstorage” on page 18-7.

To run **dbcc checkstorage** and one of the system procedures for generating reports with a single command, use **sp_dbcc_runcheck**. For information on the report generating system procedures, see "Generating Reports from dbccdb" on page 18-41.

dbcc checktable

The **dbcc checktable** command checks the specified table to see that:

- Index and data pages are linked correctly
- Indexes are sorted properly
- Pointers are consistent
- Data rows on each page have entries in the row-offset table; these entries match the locations for the data rows on the page
- data rows on each page have entries in the row-offset table in the page that matches their respective locations on the page
- Partition statistics for partitioned tables are correct

The syntax for **dbcc checktable** is:

```
dbcc checktable ({table_name | table_id}
   [, skip_ncindex] )
```

The **skip_ncindex** option allows you to skip checking the page linkage, pointers, and sort order on nonclustered indexes. The linkage and pointers of clustered indexes and data pages are essential to the integrity of your tables. You can drop and re-create nonclustered indexes if Adaptive Server reports problems with page linkage or pointers.

dbcc checktable can be used with the table name or the table's object ID. The *sysobjects* table stores this information in the *name* and *id* columns.

The following example shows a report on an undamaged table:

```
dbcc checktable(titles)
go

Checking titles
The total number of data pages in this table is 3.
Table has 18 data rows.

DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator
(SA) role.
```

If the table is partitioned, **dbcc checktable** checks data page linkage and partition statistics for each partition. For example:

```
dbcc checktable(historytab)
go

Checking historytab
The total number of pages in partition 1 is 20.
The total number of pages in partition 2 is 17.
The total number of pages in partition 3 is 19.
The total number of pages in partition 4 is 17.
The total number of pages in partition 5 is 20.
The total number of pages in partition 6 is 16.
The total number of pages in partition 7 is 19.
The total number of pages in partition 8 is 17.
The total number of pages in partition 9 is 19.
The total number of pages in partition 10 is 16.

The total number of data pages in this table is
190.
Table has 1536 data rows.

DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator
(SA) role.
```

For more information about partitions, see "Commands for Partitioning Tables" on page 17-22 in the *Performance and Tuning Guide*.

To check a table that is not in the current database, supply the database name. To check a table owned by another object, supply the owner's name. You must enclose any qualified table name in quotes. For example:

```
dbcc checktable("pubs2.newuser.testtable")
```

dbcc checktable addresses the following problems:

- If the page linkage is incorrect, **dbcc checktable** displays an error message.
- If the sort order (*sysindexes.soid*) or character set (*sysindexes.csid*) for a table with columns with *char* or *varchar* datatypes is incorrect, and the table's sort order is compatible with Adaptive Server's default sort order, **dbcc checktable** corrects the values for the table. Only the binary sort order is compatible across character sets.

➤ ***Note***

If you change sort orders, character-based user indexes are marked "read-only" and must be checked and rebuilt, if necessary. See Chapter 13, "Configuring Character Sets, Sort Orders, and Languages," for more information about changing sort orders.

- If data rows are not accounted for in the first OAM page for the object, **dbcc checktable** updates the number of rows on that page. This is not a serious problem. The built-in function **rowcnt** uses this value to provide fast row estimates in procedures like **sp_spaceused.**

Performance of **dbcc checktable** can be improved by using enhanced page fetching.

dbcc checkdb

The **dbcc checkdb** command runs the same checks as **dbcc checktable** on each table in the specified database. If you do not give a database name, **dbcc checkdb** checks the current database. **dbcc checkdb** gives similar messages to those returned by **dbcc checktable** and makes the same types of corrections.

The syntax for **dbcc checkdb** is:

```
dbcc checkdb [(database_name [, skip_ncindex]) ]
```

If you specify the optional **skip_ncindex** mode, **dbcc checkdb** does not check any of the nonclustered indexes on user tables in the database.

Checking Page Allocation

The dbcc commands that you use to check page allocation are:

- **dbcc checkalloc**
- **dbcc indexalloc**
- **dbcc tablealloc**

dbcc checkalloc

dbcc checkalloc checks the specified database to ensure that:

- All pages are correctly allocated

- Partition statistics on the allocation pages are correct
- No page is allocated that is not used
- No page is used that is not allocated

The syntax for **dbcc checkalloc** is:

```
dbcc checkalloc [(database_name [, fix | nofix] )]
```

If you do not provide a database name, **dbcc checkalloc** checks the current database.

The default option for **dbcc checkalloc** is **nofix**. When using **dbcc checkalloc** with the **nofix** option, it does not correct allocation errors.

With the **fix** option, **dbcc checkalloc** can fix all allocation errors that would otherwise be fixed by **dbcc tablealloc** and can also fix pages that remain allocated to objects that have been dropped from the database. Before you can use **dbcc checkalloc** with the **fix** option, you must put the database into single-user mode.

For details on using the **fix** and **no fix** options, see "Correcting Allocation Errors Using the fix | nofix Option" on page 18-16.

dbcc checkalloc reports the amount of space allocated and used. The output consists of a block of data for each table, including the system tables and the indexes on each table. For each table or index, it reports the number of pages and extents used. Table information is reported as either INDID=0 or INDID=1. Tables without clustered indexes have INDID=0, as shown in the example report on the *sailsdetail* table. Tables with clustered indexes have INDID=1. The report for these indexes includes information at the data level and at the index level, as shown in the example reports on *titleauthor* and *titles*. Nonclustered indexes are numbered consecutively, starting with INDID=2.

The following report on *pubs2* shows the output for the *sailsdetail*, *titleauthor*, and *titles* tables:

```
************************************************************
TABLE: salesdetail              OBJID = 144003544
INDID=0  FIRST=297       ROOT=299        SORT=0
        Data level: 0.  3 Data  Pages in 1 extents.
INDID=2  FIRST=289       ROOT=290        SORT=1
        Indid     : 2.  3 Index Pages in 1 extents.
INDID=3  FIRST=465       ROOT=466        SORT=1
        Indid     : 3.  3 Index Pages in 1 extents.
TOTAL # of extents = 3
************************************************************
TABLE: titleauthor              OBJID = 176003658
INDID=1  FIRST=433       ROOT=425        SORT=1
```

```
        Data level: 1.  1 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=305       ROOT=305        SORT=1
        Indid     : 2.  1 Index Pages in 1 extents.
INDID=3  FIRST=441       ROOT=441        SORT=1
        Indid     : 3.  1 Index Pages in 1 extents.
TOTAL # of extents = 4
************************************************************
TABLE: titles           OBJID = 208003772
INDID=1  FIRST=417       ROOT=409        SORT=1
        Data level: 1.  3 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=313       ROOT=313        SORT=1
        Indid     : 2.  1 Index Pages in 1 extents.
TOTAL # of extents = 3
************************************************************
```

dbcc indexalloc

The **dbcc indexalloc** command checks the specified index to see that:

- All pages are correctly allocated
- No page is allocated that is not used
- No page is used that is not allocated

dbcc indexalloc is an index-level version of **dbcc checkalloc**, providing the same integrity checks on an individual index. You can specify either the table name or the table's object ID (the *id* column in *sysobjects*) and the index's *indid* in *sysindexes*. **dbcc checkalloc** and **dbcc indexalloc** include the index ID's in their output.

The syntax for dbcc indexalloc is:

```
dbcc indexalloc ( {table_name | table_id }, index_id
   [, {full | optimized | fast | null}
   [, fix | nofix]])
```

If you want to use the **fix** or **nofix** option for **dbcc indexalloc**, you must also specify one of the report options (**full**, **optimized**, **fast**, or **null**). For details on using the **fix** and **no fix** options, see "Correcting Allocation Errors Using the fix | nofix Option" on page 18-16. For details on the reports, see "Generating Reports with dbcc tablealloc and dbcc indexalloc" on page 18-17.

You can run **sp_indsuspect** to check the consistency of sort order in indexes and **dbcc reindex** to repair inconsistencies. For details see "Using sp_indsuspect to Find Corrupt Indexes" on page 13-11 and "Rebuilding Indexes After Changing the Sort Order" on page 13-12.

dbcc tablealloc

dbcc tablealloc checks the specified user table to ensure that:

- All pages are correctly allocated
- Partition statistics on the allocation pages are correct
- No page is allocated that is not used
- No page is used that is not allocated

The syntax for **dbcc tablealloc** is:

```
dbcc tablealloc ({table_name | table_id}
   [, {full | optimized | fast | null}
   [, fix | nofix]])
```

You can specify either the table name or the table's object ID from the *id* column in *sysobjects*.

If you want to use the **fix** or **nofix** options for **dbcc tablealloc**, you must also specify one of the report options (**full**, **optimized**, **fast**, or **null**). For details on using the **fix** and **no fix** options, see "Correcting Allocation Errors Using the fix | nofix Option" on page 18-16. For details on the reports, see "Generating Reports with dbcc tablealloc and dbcc indexalloc" on page 18-17.

Correcting Allocation Errors Using the *fix | nofix* Option

The **fix | nofix** option can be used with **dbcc checkalloc**, **dbcc tablealloc**, and **dbcc indexalloc**. It determines whether or not the command fixes the allocation errors in tables. The default for all user tables is **fix**. The default for all system tables is **nofix**.

Before you can use the **fix** option on system tables, you must put the database into single-user mode with the following command:

```
sp_dboption dbname, "single user", true
```

You can issue this command only when no one is using the database. While it is in effect, only the user who issued it can access the database. Because of this, we recommend running **dbcc checkalloc** with the **nofix** option, so that the database is available to other users, and then using **dbcc tablealloc** or **dbcc indexalloc** with the **fix** option to correct errors in individual tables or indexes.

➤ ***Note***

The number of errors corrected when you run **dbcc tablealloc** and **dbcc indexalloc** with the **fix** option depends on the type of report (**full**, **optimized**, or **fast**) you request. Running the commands with the **full** option fixes more errors than running them with the **optimized** or **fast** option. For details on running reports for these commands, see "Generating Reports with dbcc tablealloc and dbcc indexalloc" on page 18-17.

Output from a **dbcc tablealloc** command with the **fix** option displays allocation errors, as well as any corrections that were made. The following example shows an error message that appears whether or not the **fix** option is used:

```
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt: The entry is missing from the OAM
for object id 144003544 indid 0 for allocation
page 2560.
```

When the **fix** option is used, the following message indicates that the missing entry has been restored:

```
The missing OAM entry has been inserted.
```

The **fix|nofix** option works the same in **dbcc indexalloc** as it does in **dbcc tablealloc.**

Generating Reports with *dbcc tablealloc* and *dbcc indexalloc*

You can generate three types of reports with **dbcc tablealloc** or **dbcc indexalloc**:

- **full** – Produces a report containing all types of allocation errors. Using the **full** option with **dbcc tablealloc** gives the same results as using **dbcc checkalloc** at a table level.
- **optimized** – Produces a report based on the allocation pages listed in the OAM pages for the table. When you use the **optimized** option, **dbcc tablealloc** does not report and cannot fix unreferenced extents on allocation pages that are not listed in the OAM pages. If the type of report is not specified in the command, or if you specify **null**, **optimized** is the default.
- **fast** – Produces an exception report of pages that are referenced but not allocated in the extent (2521-level errors). An allocation report is not produced.

For a comparison of speed, completeness, locking, and performance issues for these options and other **dbcc** commands, see Table 18-2 on page 18-19.

Checking Consistency of System Tables

The **dbcc checkcatalog** command checks for consistency within and between the system tables found in a database. For example, it verifies that:

- Every type in *syscolumns* has a matching entry in *systypes*
- Every table and view in *sysobjects* has at least one column in *syscolumns*
- The last checkpoint in *syslogs* is valid

It also lists the segments defined for use by the database.

The syntax for **dbcc checkcatalog** is:

```
dbcc checkcatalog [(database_name)]
```

If you do not specify a database name, **dbcc checkcatalog** checks the current database.The following example shows output from **dbcc checkcatalog**:

```
dbcc checkcatalog (testdb)
```

```
Checking testdb
The following segments have been defined for database 5
(database name testdb).
virtual start addr      size      segments
--------------------    ------    -------------------------
33554432                4096       0
                                   1
16777216                102        2
DBCC execution completed. If DBCC printed error messages, see
your System Administrator.
```

Strategies for Using Consistency Checking Commands

The following sections compare the performance of the **dbcc** commands, provide suggestions for scheduling and strategies to avoid serious performance impacts, and provide information about **dbcc** output.

Comparing the Performance of *dbcc* Commands

Table 18-2 compares the speed, thoroughness, the level of checking and locking, and other performance implications of the **dbcc** commands. When planning which commands to use, remember that **dbcc checkdb**, **dbcc checktable**, and **dbcc checkcatalog** perform different types of integrity checks than **dbcc checkalloc**, **dbcc tablealloc**, and **dbcc indexalloc**. **dbcc checkstorage** performs a combination of the some of the checks performed by the other commands. Table 18-1 on page 18-6 shows which checks are performed by the commands.

Table 18-2: Comparison of the performance of dbcc commands

Command and Option	Level	Locking and Performance	Speed	Thoroughness
checkstorage	Page chains and data rows for all indexes, allocation pages, OAM pages, device and partition statistics	No locking; performs extensive I/O and may saturate the system's I/O; can use dedicated cache with minimal impact on other caches	Fast	High
checktable **checkdb**	Page chains, sort order, data rows, and partition statistics for all indexes	Shared table lock; **dbcc checkdb** locks one table at a time and releases the lock after it finishes checking that table	Slow	High
checktable **checkdb** with **skip_ncindex**	Page chains, sort order, and data rows for tables and clustered indexes	Shared table lock; **dbcc checkdb** locks one table at a time and releases the lock after it finishes checking that table	Up to 40% faster than without the **skip_ncindex** option	Medium
checkalloc	Page chains and partition statistics	No locking; performs extensive I/O and may saturate the I/O calls; only allocation pages are cached	Slow	High
tablealloc full **indexalloc full** with **full**	Page chains	Shared table lock; performs extensive I/O; only allocation pages are cached	Slow	High
tablealloc **indexalloc** with **optimized**	Allocation pages	Shared table lock; performs extensive I/O; only allocation pages are cached	Moderate	Medium

Table 18-2: Comparison of the performance of dbcc commands (continued)

Command and Option	Level	Locking and Performance	Speed	Thoroughness
tablealloc **indexalloc** with **fast**	OAM pages	Shared table lock	Fast	Low
checkcatalog	Rows in system tables	Shared page locks on system catalogs; releases lock after each page is checked; very few pages cached	Moderate	Medium

Using Large I/O and Asynchronous Prefetch

Some **dbcc** commands can use large I/O and asynchronous prefetch when these are configured for the caches used by the databases or objects to be checked.

dbcc checkdb and **dbcc checktable** use large I/O pools for the page chain checks on tables when the tables use a cache with large I/O configured. The largest I/O size available is used. When checking indexes, **dbcc** uses only 2K buffers.

The **dbcc checkdb**, **dbcc checktable**, and the **dbcc** allocation checking commands, **checkalloc**, **tablealloc** and **indexalloc**, use asynchronous prefetch when it is available for the pool in use. See "Setting Asynchronous Prefetch Limits for dbcc" on page 18-14 in the *Performance and Tuning Guide* for more information.

Cache binding commands and the commands to change the size and asynchronous prefetch percentages for pools are dynamic commands. If you use these **dbcc** commands during off-peak periods, when user applications experience little impact, you can change these settings to speed **dbcc** performance and restore the normal settings when **dbcc** checks are finished. See Chapter 9, "Configuring Data Caches," for information on these commands.

Scheduling Database Maintenance at Your Site

The following sections describe several factors that determine how often you should run **dbcc** commands and which ones you need to run.

Database Use

When are your databases used heavily? Is your installation used heavily primarily between the hours of 8:00 a.m. and 5:00 p.m. or is it used heavily 24 hours a day?

If your Adaptive Server is used primarily between the hours of 8:00 a.m. and 5:00 p.m., Monday through Friday, you can run **dbcc** checks at night and on weekends so that the checks do not have a significant impact on your users. If your tables are not extremely large, you can run a complete set of **dbcc** commands fairly frequently.

dbcc checkstorage and **dbcc checkcatalog** provide the best coverage at the lowest cost, and provide sufficient coverage to assure recovery from backups. **dbcc checkdb** or **dbcc checktable** can be run less frequently to check index sort order and consistency. This check does not need to be coordinated with any other database maintenance activity. Reserve the object-level **dbcc** checks and those checks that use the **fix** option for further diagnosis and correction of faults found by **dbcc checkstorage**.

If your Adaptive Server is used 24 hours a day, 7 days a week, you may want to limit the resource usage of **dbcc checkstorage** by limiting the number of worker processes or by using application queues. If you decide not to use **dbcc checkstorage**, you may want to schedule a cycle of checks on individual tables and indexes using **dbcc checktable**, **dbcc tablealloc**, and **dbcc indexalloc**. At the end of the cycle, when all tables have been checked, you can run **dbcc checkcatalog** and back up the database. For information on using application queues, see Chapter 22, “Distributing Engine Resources Between Tasks,” in the *Performance and Tuning Guide*.

Some sites with 24-hour, high-performance demands run **dbcc** checks by:

- Dumping the database to tape
- Loading the database dump into a separate Adaptive Server to create a duplicate database
- Running **dbcc** commands on the duplicate database
- Running **dbcc** commands with the **fix** options on appropriate objects in the original database, if errors are detected that can be repaired with the **fix** options

The dump is a logical copy of the database pages; therefore, problems found in the original database are present in the duplicate database. This strategy is useful if you are using dumps to provide a duplicate database for reporting or some other purpose.

Schedule the use of **dbcc** commands that lock objects to avoid interference with business activities. For example, **dbcc checkdb** acquires locks on all objects in the database while it performs the check. You cannot control the order in which it checks the objects. If you are running an application that uses *table4*, *table5*, and *table6*, and running **dbcc checkdb** takes 20 minutes to complete, the application will be blocked from accessing these tables, even when the command is not checking them.

Backup Schedule

How often do you back up your databases with **dump database**?

The more often you back up your databases and dump your transaction logs, the more data you can recover in case of failure. You and the users at your site must decide how much data you are willing to lose in the event of a disaster and develop a dump schedule to support that decision.

After you schedule your dumps, decide how to incorporate the **dbcc** commands into that schedule. It is not mandatory that you perform **dbcc** checks before each dump. However, you may lose additional data if a corruption occurs in the dumps.

An ideal time to dump a database is after you run a complete check of that database using **dbcc checkstorage** and **dbcc checkcatalog**. If these commands find no errors in the database, you know that your backup contains a clean database. Problems that occur after loading the dump can be corrected by reindexing. Use **dbcc tablealloc** or **indexalloc** on individual tables and indexes to correct allocation errors reported by **dbcc checkalloc**.

Size of Tables and Importance of Data

Answer the following questions about your data:

- How many tables contain highly critical data?
- How often does that data change?
- How large are those tables?

  dbcc checkstorage is a database-level operation. If only a few tables contain critical data or data that changes often, you may want to run the table- and index-level **dbcc** commands more frequently on those tables than you run **dbcc checkstorage** on the entire database.

Understanding the Output from *dbcc* commands

The **dbcc checkstorage** command does not display the results of the check on your computer screen. Instead, the results are stored in the *dbccdb* database. You can print a variety of reports from this database. For details on **dbcc checkstorage**, see "dbcc checkstorage" on page 18-7.

The output of most other **dbcc** commands includes information that identifies the object or objects being checked and error messages that indicate what problems, if any, the command finds in the object. When **dbcc tablealloc** and **dbcc indexalloc** are run with the **fix** option, the output also indicates the repairs that the command makes.

The following example shows **dbcc tablealloc** output for a table with an allocation error:

```
dbcc tablealloc(table5)
```

Information from *sysindexes* about the object being checked:

```
TABLE: table5            OBJID = 144003544
INDID=0  FIRST=337       ROOT=2587       SORT=0
```

Error message:

```
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt: The entry is missing from the OAM for object id
144003544 indid 0 for allocation page 2560.
```

Message indicating that the error has been corrected:

```
The missing OAM entry has been inserted.
        Data level: 0.  67 Data  Pages in 9 extents.
```

***dbcc* report on page allocation:**

```
TOTAL # of extents = 9
Alloc page 256 (# of extent=1 used pages=8 ref pages=8)
EXTID:560 (Alloc page: 512) is initialized.  Extent follows:
NEXT=0 PREV=0 OBJID=144003544 ALLOC=0xff DEALL=0x0 INDID=0 STATUS=0x0
Alloc page 512 (# of extent=2 used pages=8 ref pages=8)
Page 864 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x1)
Page 865 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x3)
Page 866 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x7)
Page 867 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0xf)
Page 868 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x1f)
Page 869 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x3f)
Page 870 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x7f)
Page 871 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0xff)
Alloc page 768 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1024 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1280 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1536 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1792 (# of extent=1 used pages=8 ref pages=8)
Alloc page 2048 (# of extent=1 used pages=8 ref pages=8)
```

(Other output deleted.)

Information on resources used:

```
Statistical information for this run follows:

Total # of pages read = 68
Total # of pages found cache = 68
Total # of physical reads = 0
Total # of saved I/O = 0
```

Message printed on completion of *dbcc* command:

```
DBCC execution completed. If DBCC printed error messages, contact a user
with System Administrator (SA) role.
```

Errors Generated by Database Consistency Problems

Errors generated by database consistency problems encountered by **dbcc checkstorage** are documented in the *dbcc_types* table. Most are in the ranges 5010–5019 and 100,000–100,032. For information on specific errors, see “dbcc_types” on page 19-6. **dbcc checkstorage** records two kinds of faults: soft and hard. For information on these faults, see “Comparison of Soft and Hard Faults” on page 18-25.

Errors generated by database consistency problems encountered by **dbcc** commands other than **dbcc checkstorage** usually have error numbers from 2500 to 2599 or from 7900 to 7999. These messages, and others that can result from database consistency problems (such

as Error 605), may include phrases like "Table Corrupt" or "Extent not within segment."

Some messages indicate severe database consistency problems; others are not so urgent. A few may require help from Sybase Technical Support, but most can be solved by:

- Running **dbcc** commands that use the **fix** option
- Following the instructions in the *Troubleshooting Guide*, which contains step-by-step instructions for resolving many database errors found by **dbcc**.

Whatever techniques are required to solve the problems, the solutions are much easier when you find the problem soon after the occurrence of the corruption or inconsistency. Consistency problems can exist on data pages that are not used frequently, such as a table that is updated only monthly. **dbcc** can find, and often fix, these problems for you.

Comparison of Soft and Hard Faults

When **dbcc checkstorage** finds a fault in the target database, it records the fault in the *dbcc_faults* table as either a **soft fault** or a **hard fault**. The following sections describe the two kinds of faults.

Soft Faults

A **soft fault** is an inconsistency in Adaptive Server that is usually not persistent. Most soft faults result from temporary inconsistencies in the target database caused by users updating the target database during the **dbcc checkstorage** operation or by **dbcc checkstorage** encountering Data Definition Language (DDL) commands. These faults are not repeated when you run the command a second time. Soft faults can by reclassified by comparing the results of the two executions of **dbcc checkstorage** or by running **dbcc tablealloc** and **dbcc checktable** after **dbcc checkstorage** finds soft faults.

If the same soft faults occur in successive executions of **dbcc checkstorage**, they can be considered "persistent" soft faults. Persistent soft faults may indicate a corruption. If **dbcc checkstorage** is executed in single-user mode, the soft faults reported are "persistent" soft faults. These faults can be resolved by using **sp_dbcc_differentialreport** or by running **dbcc tablealloc** and **dbcc checktable**. If you use the latter two commands, you need to check only the tables or indexes that exhibited the soft faults.

Hard Faults

A **hard fault** is a persistent corruption of Adaptive Server. It cannot be corrected by restarting Adaptive Server. Not all hard faults are equally severe. For example, all the following situations cause a hard fault, but the results are different:

- A page that is allocated to a nonexistent table minimally reduces the available disk storage.
- A table with some rows that are unreachable by a scan might return the wrong results.
- A table that is linked to another table causes the query to stop.

Some hard faults can be corrected by simple actions such as truncating the affected table. Others can be corrected only by restoring the database from a backup.

Dropping a Damaged Database

Use the **dbcc dbrepair dropdb** command to drop a damaged database. The Transact-SQL command **drop database** does not work on a database that cannot be recovered or used. Run the **dbrepair** command from the *master* database. No users, including the user running **dbrepair**, can be using the database when it is dropped.

The syntax for **dbcc dbrepair** is:

```
dbcc dbrepair (database_name, dropdb )
```

Preparing to Use *dbcc checkstorage*

Before you can use **dbcc checkstorage**, you must configure Adaptive Server and set up the *dbccdb* database. Table 18-3 summarizes the steps and commands in the order you should use them. Each action is described in detail in the following sections.

➤ ***Note***

Do not attempt to perform the actions or use the commands in Table 18-3 before reading the information in the recommended section. You must understand the consequences of each action on the performance of Adaptive Server and **dbcc checkstorage** before you make any changes.

Table 18-3: Tasks for preparing to use *dbcc checkstorage*

For This Action	See	Use This Command
1. Obtain recommendations for database size, devices (if *dbccdb* does not exist), workspace sizes, cache size, and the number of worker processes for the target database.	"Planning Resources" on page 18-28 "Planning Workspace Size" on page 18-30	**use master** **sp_plan_dbccdb**
2. If necessary, adjust the number of worker processes that Adaptive Server uses.	"Configuring Worker Processes" on page 18-31	**sp_configure** **number of worker processes** **memory per worker process**
3. Create a **dbcc** named cache (optional).	"Setting Up a dbcc Named Cache" on page 18-33	**sp_cacheconfig**
4. Configure a 16K I/O buffer pool.	"Configuring a 16K I/O buffer pool" on page 18-34	**sp_poolconfig**
5. If *dbccdb* already exists, drop it and all associated devices before creating a new *dbccdb* database.		**drop database**
6. Initialize disk devices for the *dbccdb* data and the log.	"Allocating Disk Space for dbccdb" on page 18-34	**disk init**
7. Create *dbccdb* on the data disk device.		**create database**
8. Add disk segments (optional).	"Segments for Workspaces" on page 18-35	**use dbccdb** **sp_addsegment**
9. Populate the *dbccdb* database and install **dbcc** stored procedures.		**isql -Usa -P -i $SYBASE/scripts/installdbccdb**

Table 18-3: Tasks for preparing to use *dbcc checkstorage (continued)*

For This Action	See	Use This Command
10. Create the workspaces.	"dbccdb Workspaces" on page 19-13	**sp_dbcc_createws**
11. Update configuration values.	"Updating the dbcc_config Table" on page 18-38	**sp_dbcc_updateconfig** **max worker processes** **dbcc named cache** **scan workspace** **text workspace** **OAM count threshold** **IO error abort** **linkage error abort**

Planning Resources

Selecting the appropriate device and size for *dbccdb* is critical to the performance of **dbcc checkstorage** operations. The system procedure **sp_plan_dbccdb** provides different configuration recommendations or facts for the specified target database depending on whether *dbccdb* exists or not. You use this information to configure Adaptive Server and set up the *dbccdb* database.

Examples of *sp_plan_dbccdb* Output

If *dbccdb* does not exist, **sp_plan_dbccdb** returns the following information:

- Minimum size for *dbccdb*
- Devices that are suitable for *dbccdb*
- Minimum sizes for the *scan* and *text* workspaces
- Minimum cache size
- Number of worker processes

The values recommended for the cache size are approximate because the optimum cache size for *dbccdb* depends on the pattern of the page allocation in the target database. The following example shows the output of **sp_plan_dbccdb** for the *pubs2* database when *dbccdb* does not exist:

```
sp_plan_dbccdb pubs2
```

```
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process
count are:

dbname        scan ws    text ws    cache    process count

pubs2         64K         64K        640K     1
```

If *dbccdb* already exists, **sp_plan_dbccdb** returns the following information:

- Minimum size for *dbccdb*
- Size of existing dbccdb database
- Minimum sizes for the *scan* and *text* workspaces
- Minimum cache size
- Number of worker processes

The following example shows the output of **sp_plan_dbccdb** for the *pubs2* database when *dbccdb* already exists:

```
sp_plan_dbccdb pubs2
```

```
Recommended size for dbccdb database is 23MB (data = 21MB, log = 2MB).

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process count are:

dbname                          scan ws    text ws    cache    process
count

pubs2                            64K        48K        640K     1
```

If you plan to check more than one database, use the name of the largest one for the target database. If you do not provide a target database name, **sp_plan_dbccdb** returns configuration values for all databases listed in *master..sysdatabases*, as shown in the following example:

```
                sp_plan_dbccdb
Recommended size for dbccdb is 4MB.

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process
count are:

dbname           scan ws    text ws    cache    process count

master            64K        64K        640K     1
tempdb            64K        64K        640K     1
model             64K        64K        640K     1
sybsystemprocs   384K       112K       1280K     2
pubs2             64K        64K        640K     1
pubs3             64K        64K        640K     1
pubtune          160K        96K       1280K     2
sybsecurity       96K        96K       1280K     2
dbccdb           112K        96K      1280K      2
```

For more information, see **sp_plan_dbccdb** in the *Adaptive Server Reference Manual*.

Planning Workspace Size

Two workspaces are required for *dbccdb*: *scan* and *text*. Space requirements for the workspaces depend on the size of the largest database that will be checked. Additional workspaces are necessary if you want to run concurrent **dbcc checkstorage** operations.

Determining the SIze for the Largest Database to be Checked

Different databases can use the same workspaces. Therefore, the workspaces must be large enough to accommodate the largest database with which they will be used.

➤ ***Note***

sp_plan_dbccdb suggests workspace sizes, so you need not compute the workspace size using the formula described in the following paragraphs. The details for determining the workspace size are provided for background information only.

Each page in the **target database** is represented by one 18-byte row in the *scan* workspace. This workspace should be approximately 1.1 percent of the target database size.

If there are *text* columns in the target database, every non-null *text* column inserts a 22-byte row in the *text* workspace. If there are *n* non-null *text* columns in the target database, the size of the *text* workspace must be at least (22 * *n*) bytes. The number of non-null *text* columns is dynamic, so be sure to allocate enough space for the *text* workspace to accommodate future demands. The minimum space required for the *text* workspace is 24 pages.

Number of Workspaces That Can Be Used Concurrently

You can set up *dbccdb* configuration so that you can run **dbcc checkstorage** concurrently on multiple databases. This is possible only when the second and subsequent **dbcc checkstorage** operations have their own dedicated resources. If you want to perform concurrent **dbcc checkstorage** operations, each operation must have its own *scan* and *text* workspaces, worker processes, and reserved cache.

The total space requirement for workspaces depends on whether the user databases are checked serially or concurrently. If **dbcc checkstorage** operations are run serially, the largest *scan* and *text* workspaces can be used for all user databases. If **dbcc checkstorage** operations are run concurrently, then *dbccdb* should be set to accommodate the largest workspaces that will be used concurrently. You can determine the workspace sizes by adding the sizes of the largest databases that will be checked concurrently.

For more information on the *dbccdb* workspaces, see “dbccdb Workspaces” on page 19-13.

Configuring Adaptive Server for *dbcc checkstorage*

The following sections provide information on configuring Adaptive Server for **dbcc checkstorage.**

Configuring Worker Processes

The following parameters affect **dbcc checkstorage**:

- **max worker processes** – Set this **dbcc** parameter with **sp_dbcc_updateconfig.** It updates the value of *max worker processes* in the *dbcc_config* table for each target database.

- **number of worker processes** – Set this configuration parameter with **sp_configure**. It updates the *server_name.cfg* file.
- **memory per worker process** – Set this configuration parameter with **sp_configure**. It updates the *server_name.cfg* file.

After changing the value of the **sp_configure** parameters, you must restart Adaptive Server for the change to take effect. For details, see Chapter 11, "Setting Configuration Parameters."

The **max worker processes** parameter specifies the maximum number of worker processes used by **dbcc checkstorage** for each target database. The **number of worker processes** parameter specifies the total number of worker processes supported by Adaptive Server. The worker processes are not dedicated to running **dbcc checkstorage** operations.

Set the value for **number of worker processes** high enough to allow for the sum of the number of processes specified by **max worker processes**. A low number of worker processes reduces the performance and resource consumption of **dbcc checkstorage**. The number of worker processes should not exceed the number of database devices used by the database. **dbcc checkstorage** will not use more than that number. Cache size, CPU performance, and device sizes might suggest a lower worker processes count. If there are not enough worker processes configured for Adaptive Server, **dbcc checkstorage** will not run.

The configuration parameters **maximum parallel degree** and **maximum scan parallel degree** have no effect on the parallel functions of **dbcc checkstorage**. When **maximum parallel degree** is set to 1, parallelism in **dbcc checkstorage** is not disabled.

dbcc checkstorage requires multiple processes, so **number of worker processes** must be set to at least 1 to allow for a parent process and a worker process.

sp_plan_dbccdb recommends values for the number of worker processes, depending on database size, number of devices, and other factors. You can use smaller values to limit the load on your system. **dbcc checkstorage** may use fewer worker processes than **sp_plan_dbccdb** recommends or fewer than you configure.

Using more worker processes does not guarantee faster performance. The following scenario describes the effects of two different worker process configurations:

An 8GB database has 4GB of data on disk A and 0.5GB of data on each of the disks B, C, D, E, F, G, H, and I.

With 9 worker processes active, the time it takes to run **dbcc checkstorage** is 2 hours, which is the time it takes to check disk A. Each of the other 8 worker processes finishes in 15 minutes and waits for the disk A worker process to finish.

With 2 worker processes active, the time it takes to run **dbcc checkstorage** is still 2 hours. The first worker process processes disk A and the other worker process processes disks B, C, D, E, F, G, H, and I. In this case, there is no waiting, and resources are used more efficiently.

The **memory per worker process** parameter specifies the total memory allocation for worker processes support in Adaptive Server. The default value is adequate for **dbcc checkstorage**.

Setting Up a *dbcc* Named Cache

If you use a named cache for **dbcc checkstorage**, you might need to adjust the Adaptive Server configuration parameters.

During a **dbcc checkstorage** operation, the workspaces are temporarily bound to a cache. This cache is also used to read the **target database**. Using a named cache that is dedicated to **dbcc** minimizes the impact of the database check on other users and improves performance. You can create a separate cache for each **dbcc checkstorage** operation that will be run concurrently, or you can create one cache that is large enough to fit the total requirements of the concurrent operations. The size of the named cache required for optimum performance depends on the size of the target database and distributions of data in that database. **dbcc checkstorage** requires a minimum of 640K of 16K buffers per worker process in the named cache.

For best performance, assign most of the dedicated cache to the 16K buffer pool. The recommended cache size is the minimum size for the 16K pool. Add the size of the 2K pool to this value.

If you dedicate a cache for **dbcc checkstorage**, the command does not require more than the minimum 2K buffer pool (0.5MB). If the cache is shared, you can improve the performance of **dbcc checkstorage** by increasing the 16K pool size before running the operation, and reducing the size after the operation is complete. The 16K pool requirements are the same for a shared cache. However, while a shared cache may meet the size requirement, other demands on the cache might limit the buffer availability to **dbcc checkstorage** and greatly impact the performance of both **checkstorage** and Adaptive Server as a whole.

To configure Adaptive Server with a named cache for **dbcc checkstorage** operations, use **sp_cacheconfig** and **sp_poolconfig**. For details, see Chapter 9, "Configuring Data Caches."

Configuring a 16K I/O buffer pool

dbcc checkstorage requires a 16K I/O buffer pool to be configured. You use **sp_poolconfig** to configure the pool size and to see whether the pool has been configured properly. The information about the pool size is stored in the *dbcc_config* table.

The following example shows how to use **sp_poolconfig** to set the 16K buffer pool for "master_cache", the named cache that was created for the *master* database.

```
1> sp_poolconfig "master_cache", "1024K", "16K"
2> go
(return status = 0)
```

The following example shows that the buffer pool for the private cache "master_cache" is set:

```
1> sp_poolconfig "master_cache"
2> go
Cache Name     Status    Type     Config Value Run Value
-------------  --------- -------- ------------ ------------
master_cache   Active    Mixed         2.00 Mb      2.00 Mb
                                    ------------ ------------
                         Total        2.00 Mb      2.00 Mb
===============================================================
Cache: master_cache,   Status: Active,   Type: Mixed
   Config Size: 2.00 Mb,   Run Size: 2.00 Mb
   Config Replacement: strict LRU,   Run Replacement: strict LRU
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
2 Kb     512 Kb    0.00 Mb      1.00 Mb      10
16 Kb    192 Kb    1.00 Mb      1.00 Mb      10
(return status = 0)
```

For more information on **sp_poolconfig**, see the *Adaptive Server Reference Manual*.

Allocating Disk Space for *dbccdb*

Additional disk storage is required for the *dbccdb* database. Because **dbcc checkstorage** uses *dbccdb* extensively, you should place *dbccdb* on a device that is separate from other database devices. To maximize

performance and minimize the impact of hardware-induced corruptions, allocate *dbccdb* on a dedicated disk whenever possible.

➤ ***Note***

Do not create *dbccdb* on the master device. Make sure that the log devices and data devices for *dbccdb* are separate.

Segments for Workspaces

By dedicating segments for the workspaces, you can control the placement of the workspaces and improve the performance of **dbcc checkstorage**. When you add new segments to *dbccdb* for the exclusive use of workspaces, be sure to unmap the devices attached to these segments from the default segment with **sp_dropsegment**.

Creating the *dbccdb* Database

To create the *dbccdb* database:

1. If you have not yet done so, run **sp_plan_dbccdb** in the *master* database to obtain recommendations for database size, devices (if *dbccdb* does not exist), workspace sizes, cache size, and the number of worker processes for the **target database**. For example, suppose you run **sp_plan_dbccdb** with *pubs2* as the target database when *dbccdb* did not exist, as follows:

   ```
   use master
   go

   sp_plan_dbccdb pubs2
   go
   ```

   And the following output appears:

```
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process
count are:

dbname        scan ws    text ws    cache    process count

pubs2         64K         64K        640K     1
```

For details on the information provided by **sp_plan_dbccdb**, see "Planning Resources" on page 18-28.

2. If *dbccdb* already exists, drop it and all associated devices before creating a new *dbccdb* database:

```
use master
go

if exists (select * from master.dbo.sysdatabases
     where name = "dbccdb")
begin
     print "+++ Dropping the dbccdb database"
     drop database dbccdb
end
go
```

3. Use **disk init** to initialize disk devices for the *dbccdb* data and the log:

```
use master
go

disk init
   name = "dbccdb_dat",
   physname = "/remote/disks/masters/",
   vdevno = 4,
   size = 4096
go
```

```
disk init
   name = "dbccdb_log",
   physname = "/remote/disks/masters/",
   vdevno = 5,
   size = 1024
go
```

4. Use **create database** to create *dbccdb* on the data disk device that you initialized in step 3:

   ```
   use master
   go

   create database dbccdb
      on dbccdb_dat = 6
      log on dbccdb_log = 2
   go
   ```

5. (Optional) Use **sp_addsegment** to add segments for the *scan* and *text* workspaces to the *dbccdb* data device:

   ```
   use dbccdb
   go

   sp_addsegment scanseg, dbccdb, dbccdb_dat
   go

   sp_addsegment textseg, dbccdb, dbccdb_dat
   go
   ```

6. Use the script **installdbccdb** to create the tables for *dbccdb* and to initialize the *dbcc_types* table:

   ```
   isql -Ujms -P***** -iinstalldbccdb
   ```

   The script checks for the existence of the database before attempting to create the tables. It creates only those tables that do not already exist in *dbccdb*. If any of the *dbccdb* tables become corrupted, remove them with **drop table**, and then use **installdbccdb** to re-create them.

7. Use **sp_dbcc_createws** to create and initialize the *scan* and *text* workspaces:

   ```
   use dbccdb
   go

   sp_dbcc_createws dbccdb, scanseg, scan_pubs2,
   scan, "10M"
   go

   sp_dbcc_createws dbccdb, textseg, text_pubs2,
   text, "10M"
   go
   ```

When you have finished installing *dbccdb*, you must update the *dbcc_config* table.

Updating the *dbcc_config* Table

Use **sp_dbcc_updateconfig** to initialize the *dbcc_config* table for the **target database**. You must update each **dbcc** parameter separately for each target database, as shown in the following example.

```
use dbccdb
go

sp_dbcc_updateconfig pubs2,"max worker processes", "4"
go

sp_dbcc_updateconfig pubs2, "dbcc named cache", pubs2_cache, "10K"
go

sp_dbcc_updateconfig pubs2, "scan workspace", scan_pubs2
go

sp_dbcc_updateconfig pubs2, "text workspace", text_pubs2
go

sp_dbcc_updateconfig pubs2, "OAM count threshold", "5"
go

sp_dbcc_updateconfig pubs2, "IO error abort", "3"
go

sp_dbcc_updateconfig pubs2,"linkage error abort", "8"
go
```

You have completed all the steps required to set up the *dbccdb* database. You can now use **dbcc checkstorage** to check your databases. For descriptions of the **dbcc** parameters, see *type code* values 1 through 9 in "dbcc_types" on page 19-6.

Maintaining *dbccdb*

You will occasionally need to perform the following maintenance tasks on *dbccdb*:

- Reevaluate and update the configuration using the following system procedures:

  sp_dbcc_evaluatedb – recommends values for configuration parameters using the results of previous **dbcc checkstorage** operations.

sp_dbcc_updateconfig – updates the configuration parameters for the specified database.

- Clean up obsolete data in *dbccdb* using the following system procedures:

  sp_dbcc_deletedb – deletes all the information on the specified database from *dbccdb*.

  sp_dbcc_deletehistory – deletes the results of the **dbcc checkstorage** operations on the specified database from *dbccdb*.

- Perform consistency checks on *dbccdb* itself.

The following sections describe the maintenance tasks in greater detail.

Reevaluating and Updating *dbccdb* Configuration

If the characteristics of user databases change, use the stored procedure **sp_dbcc_evaluatedb** to reevaluate the current *dbccdb* configuration and recommend more suitable values.

The following changes to user databases might affect the *dbccdb* configuration, as follows:

- When a user database is created, deleted or altered, the size of the workspaces and named cache, or the number of worker threads stored in the *dbcc_config* table might be affected.
- Changes in the named cache size or worker process count for **dbcc_checkstorage** might require you to reconfigure the Adaptive Server buffer cache and worker processes.

If the results of **dbcc checkstorage** operations are available for the **target database**, use **sp_dbcc_evaluatedb** to determine new configuration values. **sp_dbcc_evaluatedb** assumes the existence of *dbccdb* and expects the results of **dbcc checkstorage** operations on the target database to be stored in *dbccdb*. If *dbccdb* does not exist or if the results of **dbcc checkstorage** operations for the target database are not available, **sp_dbcc_evaluatedb** reports this fact, instead of configuration parameters. **sp_dbcc_configreport** also reports the configuration parameters for the specified database.

Use **sp_dbcc_updateconfig** to add new databases to the *dbcc_config* table and to change the configuration values in *dbcc_config* to reflect the values recommended by **sp_dbcc_evaluatedb**.

Cleaning Up *dbccdb*

Each time you run **dbcc checkstorage**, Adaptive Server stores the data generated from that operation in *dbccdb*. You should periodically clean up *dbccdb* by using **sp_dbcc_deletehistory** to delete data for the target database that was created before the date that you specify.

When a database is deleted, all configuration information and **dbcc checkstorage** results related to that database should be deleted from *dbccdb*. Use **sp_dbcc_deletedb** to delete all database information from *dbccdb*.

Performing Consistency Checks on *dbccdb*

The limited update activity in the *dbccdb* tables should make corruption less frequent. Two signs of corruption in *dbccdb* are:

- Failure of **dbcc checkstorage** during the initialization phase, as it evaluates the work that needs to be performed, or during the completion phase, when it records its results
- Loss of information about faults resulting from corruption in the recorded faults, found by **dbcc checkstorage**

dbcc checkstorage can be used to check *dbccdb*, using *dbccdb* to store the results; however, a severe corruption in *dbccdb* may cause **dbcc checkstorage** to fail during this check. To permit **dbcc checkstorage** to locate severe corruptions in *dbccdb*, you can create the alternate database, *dbccalt*, which you use only for checking *dbccdb*. You create *dbccalt* using the same process that you used to create *dbccdb* as described in "Preparing to Use dbcc checkstorage" on page 18-27.

If no free devices are available for *dbccalt*, any device that is not used by the *master* database or *dbccdb* can be used.

dbcc checkstorage and the **dbcc** system procedures function the same with *dbccalt* as they do with *dbccdb*. When the target database is *dbccdb*, **dbcc checkstorage** uses *dbccalt*, if it exists. If *dbccalt* does not exist, *dbccdb* can be checked using itself as the management database. If the target database is *dbccdb* and *dbccalt* exists, the results of **dbcc checkstorage** operations on *dbccdb* are stored in *dbccalt*. If *dbccalt* does not exist, the results are stored in *dbccdb* itself.

Alternatively, **dbcc checkalloc** and **dbcc checktable** can be used to check *dbccdb*.

If *dbccdb* becomes corrupted, you can drop it and re-create it or load an older version from a backup. If you drop it, some of its diagnostic history will be lost.

Generating Reports from *dbccdb*

Several **dbcc** stored procedures are provided with *dbccdb* so that you can generate reports from the data within *dbccdb*.

To Report a Summary of *dbcc checkstorage* Operations

sp_dbcc_summaryreport reports all **dbcc checkstorage** operations that were completed for the specified database on or before the specified date. The following example shows output from this command:

```
sp_dbcc_summaryreport
```

```
DBCC Operation : checkstorage

Database Name      Start time           End Time   Operation ID
       Hard Faults Soft Faults Text Columns Abort Count
       User Name
------------------ -------------------- ---------- ------------
      ----------- ----------- ------------ -----------
      ------------------------------
sybsystemprocs     05/12/1997 10:54:45  10:54:53              1
                0           0            0           0
       sa
sybsystemprocs     05/12/1997 11:14:10  11:14:19              2
                 0           0            0           0
       sa
```

For details on these procedures, see "dbcc Stored Procedures" in the *Adaptive Server Reference Manual*.

To Report Configuration, Statistics and Fault Information

sp_dbcc_fullreport runs the following reports in the order shown:

- **sp_dbcc_summaryreport** – For an example of the output of this report, see "To Report a Summary of dbcc checkstorage Operations" on page 18-41.
- **sp_dbcc_configreport** – For an example of the output of this report, see "To See Configuration Information for a Target Database" on page 18-42.

- **sp_dbcc_statisticsreport** – For an example of the output of this report, see "To Report Statistics Information from dbcc_counter" on page 18-43.
- **sp_dbcc_faultreport short** – For an example of the output of this report, see "To Report Faults Found in a Database Object" on page 18-43.

To See Configuration Information for a Target Database

Use **sp_dbcc_configreport** to generate a report of the configuration information for a target database. The following example shows output from this command:

```
sp_dbcc_configreport
```

```
Reporting configuration information of database sybsystemprocs.

Parameter Name              Value                          Size

database name               sybsystemprocs                 51200K
dbcc named cache            default data cache             1024K
text workspace              textws_001 (id = 544004969)    128K
scan workspace              scanws_001 (id = 512004855)    1024K
max worker processes        1
operation sequence number   2
```

To Compare Results of *dbcc checkstorage* Operations

sp_dbcc_differentialreport compares the results of the **dbcc checkstorage** operations completed for the specified database object on the specified dates. The following example shows output from this command:

```
sp_dbcc_differentialreport master, sysprocedures,
checkstorage, "01/01/96", "01/02/96"
```

```
The following changes in dbcc counter values for the object
"sysprocedures" in database master have been noticed between
01/01/96 and 01/02/96.

Description        Date1   Date2

pages used          999    1020
pages reserved     1000    1024
page extent gaps     64      67
```

To Report Faults Found in a Database Object

sp_dbcc_faultreport reports faults in the specified database object that occurred on or before the specified date. You can generate a **short** or **long** report. The following example shows a short report:

```
sp_dbcc_faultreport 'short'
```

```
Database Name : sybsystemprocs

 Table Name     Index  Type Code Description         Page Number
 -------------- ------ --------- ------------------ -----------
 sysprocedures        0    100031 page not allocated        5702
 sysprocedures        1    100031 page not allocated       14151
 syslogs              0    100022 chain start error        24315
 syslogs              0    100031 page not allocated       24315
```

The following example shows part of the output of a long report. The complete report repeats the information for each object in the **target database**. The *sybsystemprocs* database was the active database at the time the command was issued.

```
sp_dbcc_faultreport 'long'
```

```
Generating 'Fault Report' for object sysprocedures in database
sybsystemprocs.

Type Code: 100031; Soft fault, possibly spurious
Page reached by the chain is not allocated.
page id:  14151
page header:
0x0000374700003788000037460000000500064 8B803EF0001000103FE0080000F
Header for 14151, next 14216, previous 14150, id = 5:1
 time stamp = 0x0001000648B8, next row = 1007, level = 0
 free offset = 1022, minlen = 15, status = 128(0x0080)
.
.
.
```

To Report Statistics Information from *dbcc_counter*

sp_dbcc_statisticsreport reports statistics information from the *dbcc_counter* table generated by **dbcc checkstorage** on or before the specified date. The following example shows output from this command:

```
sp_dbcc_statisticsreport 'sybsystemprocs',
'sysobjects'
```

```
Statistics Report on object sysobjects in database sybsystemprocs

 Parameter Name    Index Id Value
 ----------------- -------- -----------
 count             0        160.0
 max size          0        99.0
 max count         0        16.0
 bytes data        0        12829.0
 bytes used        0        15228.0
 count             1        16.0
 max size          1        9.0
 max level         1        0.0
 max count         1        16.0
 bytes data        1        64.0
 bytes used        1        176.0
 count             2        166.0
 max level         2        1.0
 max size          2        39.0
 max count         2        48.0
 bytes data        2        3092.0
 bytes used        2        4988.0


 Parameter Name         Index Id Partition Value    Dev_name
 ---------------------- -------- --------- -------- ----------------
 page gaps              0        1         16.0     master
 pages used             0        1         17.0     master
 extents used           0        1         3.0      master
 overflow pages         0        1         0.0      master
 pages overhead         0        1         1.0      master
 pages reserved         0        1         6.0      master
 page extent gaps       0        1         7.0      master
 ws buffer crosses      0        1         7.0      master
 page extent crosses    0        1         7.0      master
 page gaps              1        1         1.0      master
 pages used             1        1         2.0      master
 extents used           1        1         1.0      master
 overflow pages         1        1         0.0      master
 pages overhead         1        1         1.0      master
 pages reserved         1        1         6.0      master
 page extent gaps       1        1         0.0      master
 ws buffer crosses      1        1         0.0      master
 page extent crosses    1        1        0.0       master
```

```
page gaps              2        1        5.0      master
pages used             2        1        8.0      master
extents used           2        1        1.0      master
overflow pages         2        1        0.0      master
pages overhead         2        1        1.0      master
pages reserved         2        1        0.0      master
page extent gaps       2        1        0.0      master
ws buffer crosses      2        1        0.0      master
page extent crosses    2        1        0.0      master
```

19 *dbccdb* Tables

The **dbcc** management database, *dbccdb*, contains seven tables that define inputs to and outputs from **dbcc checkstorage**. It also contains at least two workspaces. Topics include:

- dbcc_config 19-1
- dbcc_counters 19-2
- dbcc_fault_params 19-3
- dbcc_faults 19-3
- dbcc_operation_log 19-4
- dbcc_operation_results 19-5
- dbcc_types 19-6
- dbccdb Workspaces 19-13
- dbccdb Log 19-15

dbcc_config

The *dbcc_config* table describes the currently executing or last completed **dbcc checkstorage** operation. It defines:

- The location of resources dedicated to the **dbcc checkstorage** operation
- Resource usage limits for the **dbcc checkstorage** operation

Table 19-1: Columns in the dbcc_config table

Column Name	Datatype	Description
dbid	*smallint*	Matches the *dbid* from a row in *sysdatabases*.
type_code	*int*	Matches the *type_code* from a row in the *dbcc_types* table. Valid values are 1–9.
value	*int*	Specifies the value of the item identified by *type_code*. Can be null only if the value of *stringvalue* is not null.
stringvalue	*varchar(255)*	Specifies the value of the item identified by *type_code*. Can be null only if the value of *value* is not null.

Primary key: combination of *dbid* and *type_code*

For information on initializing and updating *dbcc_config*, see "Creating the dbccdb Database" on page 18-35.

dbcc_counters

The *dbcc_counters* table stores the results of the analysis performed by **dbcc checkstorage**. Counters are maintained for each database, table, index, partition, device, and invocation of **dbcc**.

Table 19-2: Columns in the dbcc_counters table

Column Name	Datatype	Description
dbid	*smallint*	Identifies the target database.
id	*int*	Identifies the table. The value is derived from *sysindexes* and *sysobjects*.
indid	*smallint*	Identifies the index. The value is derived from *sysindexes*.
partitionid	*smallint*	Identifies the defined object-page affinity. The value is derived from *sysindexes* and *syspartitions*.
devid	*smallint*	Identifies the disk device. The value is derived from *sysdevices*.
opid	*smallint*	Identifies the **dbcc** operation that was performed.
type_code	*int*	Matches the *type_code* column of a row in the *dbcc_types* table. Valid values are 5000 through 5019.
value	*real*	Matches the appropriate *type_name* for the given *type_code* as described in *dbcc_types*.

Primary key: combination of *dbid*, *id*, *indid*, *partitionid*, *devid*, *opid*, and *type_code*

dbcc_fault_params

The *dbcc_fault_params* table provides additional descriptive information for a fault entered in the *dbcc_faults* table.

Table 19-3: Columns in the dbcc_fault_params table

Column Name	Datatype	Description
dbid	*smallint*	Identifies the target database.
opid	*smallint*	Identifies the **dbcc** operation that was performed.
faultid	*int*	Identifies the fault ID.
type_code	*int*	Defines the interpretation of the value, which is provided by the "value" columns. Valid values are 1000–1006. They are described in the *dbcc_types* table.
intvalue	*int*	Specifies the integer value.
realvalue	*real*	Specifies the real value.
binaryvalue	*varbinary(255)*	Specifies the binary value.
stringvalue	*varchar(255)*	Specifies the string value.
datevalue	*datetime*	Specifies the date value.

Primary key: combination of *dbid*, *opid*, *faultid*, and *type_code*

Each "value" column (*intvalue*, *realvalue*, *binaryvalue*, *stringvalue*, and *datevalue*) can contain a null value. At least one must not be null. If more than one of these columns contains a value other than null, the columns provide different representations of the same value.

dbcc_faults

The *dbcc_faults* table provides a description of each fault detected by **dbcc checkstorage**.

Table 19-4: Columns in the dbcc_faults table

Column Name	Datatype	Description
dbid	*smallint*	Identifies the target database.
id	*smallint*	Identifies the table. The value is derived from *sysindexes* and *sysobjects*.

Table 19-4: Columns in the dbcc_faults table (continued)

Column Name	Datatype	Description
indid	*smallint*	Identifies the index. The value is derived from *sysindexes*.
partitionid	*smallint*	Identifies the partition. The value is derived from *sysindexes* and *syspartitions*. Counters are maintained for page ranges, so "partition" refers to the defined object-page affinity, rather than the actual object page chain.
devid	*smallint*	Identifies the disk device. The value is derived from *sysdevices*
opid	*smallint*	Identifies the **dbcc** operation that was performed.
faultid	*int*	Provides a unique sequence number assigned to each fault recorded for the operation.
type_code	*int*	Identifies the type of fault. Valid values are 100000–100032. They are described in Table 19-7.
status	*int*	Classifies the fault. Valid values are: 0 = Soft fault, possibly spurious 1 = Hard fault For more information, see "Comparison of Soft and Hard Faults" on page 18-25.

Primary key: combination of *dbid*, *id*, *indid*, *partitionid*, *devid*, *opid*, *faultid*, and *type_code*

dbcc_operation_log

The *dbcc_operation_log* table records the use of the **dbcc checkstorage** operations.

Table 19-5: Columns in the dbcc_operation_log table

Column Name	Datatype	Description
dbid	*smallint*	Identifies the target database

Table 19-5: Columns in the dbcc_operation_log table (continued)

Column Name	Datatype	Description
opid	*smallint*	Identifies the sequence number of the **dbcc checkstorage** operation. *opid* is an automatically incrementing number, unique for each *dbid* and *finish* pair.
optype	*smallint*	The following value is valid for *optype*: 2 = checkstorage
suid	*int*	Identifies the user executing the command
start	*datetime*	Identifies when the operation started
finish	*datetime*	Identifies when the operation ended

Primary key: combination of *dbid*, *opid*, and *optype*

Summary results are recorded in the *dbcc_operation_results* table.

dbcc_operation_results

The *dbcc_operation_results* table provides additional descriptive information for an operation recorded in the *dbcc_operation_log* table.

Table 19-6: Columns in the dbcc_operation_results table

Column Name	Datatype	Description
dbid	*smallint*	Identifies the target database
opid	*smallint*	Identifies the **dbcc** operation ID
optype	*smallint*	Identifies the **dbcc** operation type
type_code	*int*	Defines the **dbcc** operation type. Valid values are 1000–1007. They are described in Table 19-7.
intvalue	*int*	Specifies the integer value
realvalue	*real*	Specifies the real value
binaryvalue	*varbinary(255)*	Specifies the binary value
stringvalue	*varchar(255)*	Specifies the string value
datevalue	*datetime*	Specifies the date value

Primary key: combination of *dbid*, *opid*, *optype*, and *type_code*

Each "value" column (*intvalue*, *realvalue*, *binaryvalue*, *stringvalue*, and *datevalue*) may contain a null value. At least one is not null. If more than one of these columns contains a value other than null, the columns provide different representations of the same value.

Results of the **dbcc checkstorage** operations include the number of:

- Hard faults found
- Soft faults found
- Operations stopped due to a hard error

dbcc_types

The *dbcc_types* table provides the definitions of the data types used by **dbcc checkstorage**. This table is not actually used by the **dbcc** stored procedures. It is provided to facilitate the use of the other tables in *dbccdb*, and to document the semantics of the data types. Type codes for operation configuration, analysis data reported, fault classification, and fault report parameters are included. If you create your own stored procedures for generating reports, the values listed in the *type_name* column can be used as report headings.

Table 19-7 describes the contents of the *dbcc_types* table. To allow for future additions to *dbcc_types*, some *type_code* numbers are not used at this time.

Table 19-7: Contents of the dbcc_types table

type_code	*type_name*	Description
1	max worker processes	Optional. Specifies the maximum number of worker processes that can be employed. This is also the maximum level of concurrent processing used. Minimum value is 1.
2	dbcc named cache	Specifies the size (in kilobytes) of the cache used by **dbcc checkstorage** and the name of that cache.
3	scan workspace	Specifies the ID and name of the workspace to be used by the database scan.
4	text workspace	Specifies the ID and name of the workspace to be used for text columns.
5	operation sequence number	Specifies the number that identifies the **dbcc** operation that was started most recently.
6	database name	Specifies the name of the database in *sysdatabases*.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
7	OAM count threshold	Specifies the percentage by which the OAM counts must vary before they can be consideredto be an error.
8	IO error abort	Specifies the number of I/O errors allowed on a disk before **dbcc** stops checking the pages on that disk.
9	linkage error abort	Specifies the number of linkage errors allowed before **dbcc** stops checking the page chains of an object. Some kinds of page chain corruptions might require a check to be stopped with fewer linkage error than other kinds of page chain corruptions.
1000	hard fault count	Specifies the number of persistent inconsistencies (hard faults) found during the consistency check.
1001	soft fault count	Specifies the number of suspect conditions (soft faults) found during the consistency check.
1002	checks aborted count	Specifies the number of linkage checks that were stopped during the consistency check.
1007	text column count	Specifies the number of non-null *text*/*image* column values found during the consistency check.
5000	bytes data	Specifies (in bytes) the amount of user data stored in the partition being checked.
5001	bytes used	Specifies (in bytes) the amount of storage used to record the data in the partition being checked. The difference between *bytes used* and *bytes data* shows the amount of overhead needed to store or index the data.
5002	pages used	Specifies the number of pages linked to the object being checked that are actually used to hold the object.
5003	pages reserved	Specifies the number of pages that are reserved for the object being checked, but that are not allocated for use by that object. The difference between (8 * *extents used*) and (*pages used* + *pages reserved*) shows the total uncommitted deallocations and pages incorrectly allocated.
5004	pages overhead	Specifies the number of pages used for the overhead functions such as OAM pages or index statistics.
5005	extents used	Specifies the number of extents allocated to the object in the partition being checked. For object 99 (allocation pages), this value is the number of extents that are not allocated to a valid object. Object 99 contains the storage that is not allocated to other objects.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
5006	count	Specifies the number of component items (rows or keys) found on any page in the part of the object being checked.
5007	max count	Specifies the maximum number of component items found on any page in the part of the object being checked.
5008	max size	Specifies the maximum size of any component item found on any page in the part of the object being checked.
5009	max level	Specifies the maximum number of levels in an index. This datatype is not applicable to tables.
5010	pages misallocated	Specifies the number of pages that are allocated to the object, but are not initialized correctly. It is a fault counter.
5011	io errors	Specifies the number of I/O errors encountered. This datatype is a fault counter.
5012	page format errors	Specifies the number of page format errors reported. This datatype is a fault counter.
5013	pages not allocated	Specifies the number of pages linked to the object through its chain, but not allocated. This datatype is a fault counter.
5014	pages not referenced	Specifies the number of pages allocated to the object, but not reached through its chains. This datatype is a fault counter.
5015	overflow pages	Specifies the number of overflow pages encountered. This datatype is only applicable to clustered indexes.
5016	page gaps	Specifies the number of pages not linked to the next page in ascending sequence. This number indicates the amount of table fragmentation.
5017	page extent crosses	Specifies the number of pages that are linked to pages outside of their own extent. As the number of *page extent crosses* increases relative to *pages used* or *extents used*, the effectiveness of large I/O buffers decreases.
5018	page extent gaps	Specifies the number of page extent crosses where the subsequent extent is not the next extent in ascending sequence. Maximal I/O performance on a full scan is achieved when the number of *page extent gaps* is minimized. A seek or full disk rotation is likely for each gap.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
5019	ws buffer crosses	Specifies the number of pages that are linked outside of their workspace buffer cache during the **dbcc checkstorage** operation. This information can be used to size the cache, which provides high performance without wasting resources.
10000	page id	Specifies the location in the database of the page that was being checked when the fault was detected. All localized faults include this parameter.
10001	page header	Specifies the hexadecimal representation of the header of the page that was being checked when the fault was detected. This information is useful for evaluating soft faults and for determining if the page has been updated since it was checked. The server truncates trailing zeros.
10002	text column id	Specifies an 8-byte hexadecimal value that gives the page, row, and column of the reference to a text chain that had a fault. The server truncates trailing zeros.
10003	object id	Specifies a 9-byte hexadecimal value that provides the *object id* (table), the *partition id* (partition of the table) if applicable, and the *index id* (index) of the page or allocation being checked. For example, if a page is expected to belong to table *T1* because it is reached from *T1*'s chain, but is actually allocated to table *T2*, the *object id* for *T1* is recorded, and the *object id expected* for *T2* is recorded. The server truncates trailing zeros.
10007	page id expected	Specifies the page ID that is expected for the linked page when there is a discrepancy between the page ID that is expected and the page ID that is actually encountered. For example, if you follow the chain from *P1* to *P2* when going forward, then, when going backward, *P1* is expected to come after *P2*. The value of *page id expected* is *P1*, and the value of *page id* is *P2*. When the actual value of *P3* is encountered, it is recorded as *page id actual*.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
10008	page id actual	When there is a discrepancy between the page ID that is encountered and the expected page ID, this value specifies the actual page ID that is encountered. (See also, *type_code* 10007.) For example, if you follow the chain from *P1* to *P2* when going forward, then, when going backward, P1 is expected to come after *P2*. The value of *page id expected* is *P1*, and the value of *page id* is *P2*. When the actual value of *P3* is encountered, it is recorded as *page id actual*.
10009	object id expected	Specifies a 9-byte hexadecimal value that provides the expected object id (table), the partition id (partition of the table) if applicable, and the index id (index) of the page or allocation being checked. For example, if a page is expected to belong to table *T1* because it is reached from *T1*'s chain, but is actually allocated to table *T2*, the *object id* for *T1* is recorded, and the *object id expected* for *T2* is recorded. The server truncates trailing zeros.
100000	IO error	Indicates that part of the identified page could not be fetched from the device. This is usually caused by a failure of the operating system or the hardware.
100001	page id error	Indicates that the identifying ID (page number) recorded on the page is not valid. This might be the result of a page being written to or read from the wrong disk location, corruption of a page either before or as it is being written, or allocation of a page without subsequent initialization of that page.
100002	page free offset error	Indicates that the end of data on a page is not valid. This event affects insertions and updates on this page. It might affect some access to the data on this page.
100003	page object id error	Indicates that the page appears to be allocated to some other table than the one expected. If this is a persistent fault, it might be the consequence of either: • An incorrect page allocation, which might only result in the effective loss of this page to subsequent allocation, or • A corrupted page chain, which might prevent access to the data in the corrupted chain
100004	timestamp error	Indicates that the page has a timestamp that is later than the database timestamp. This error can result in failure to recover when changes are made to this page.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
100005	wrong dbid error	Indicates that the database ID *dbid* is stored on the database allocation pages. When this ID is incorrect, the allocation page is corrupt and all the indicated allocations are suspect.
100006	wrong object error	Indicates that the page allocation is inconsistent. The page appears to belong to one table or index, but it is recorded as being allocated to some other table or index in the allocation page. This error differs from *page object id error* in that the allocation is inconsistent, but the consequences are similar.
100007	extent id error	Indicates that an allocation was found for a table or index that is unknown to **dbcc checkstorage**. Typically, this results in the inability to use the allocated storage.
100008	fixed format error	Indicates that the page incorrectly indicates that it contains only rows of a single fixed length. **dbcc checkstorage** reports this error. **dbcc checktable** does not report it, but does repair it.
100009	row format error	Indicates that at least one row on the page is incorrectly formatted. This error might cause loss of access to some or all the data on this page.
100010	row offset error	Indicates that at least one row on the page is not located at the expected page offset. This error might cause loss of access to some or all of the data on this page.
100011	text pointer error	Indicates that the location of the table row that points to the corrupted *text* or *image* data. This information might be useful for correcting the problem.
100012	wrong type error	Indicates that the page has the wrong format. For example, a data page was found in an index or a *text* / *image* column.
100013	non-OAM error	This error is a special case of *wrong type error*. It is not reported as a separate condition in the current release.
100014	reused page error	Indicates that a page is reached by more than one chain and that the chains belong to different objects. This error indicates illegal sharing of a page through corrupt page chain linkages. Access to data in either or both tables might be affected.
100015	page loop error	Indicates that a page is reached a second time while following the page chain for an object, which indicates a loop in the page chain. A loop can result in a session hanging indefinitely while accessing data in that object.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
100016	OAM ring error	Indicates that a page is allocated but not reached by the page chains for the object.Typically, this results in the inability to use the allocated storage.
100017	OAM ring error	Indicates that the OAM page ring linkages are corrupted. This might not affect access to the data for this object, but it might affect insertions, deletions, and updates to that data.
100018	missing OAM error	Indicates that **dbcc checkstorage** found an allocation for the object that was not recorded in the OAM. This error indicates a corruption that might affect future allocations of storage, but probably does not affect access to the presently stored data.
100019	extra OAM error	Indicates that an allocation for this object was recorded in the OAM, but it was not verified in the allocation page. This error indicates a corruption that might affect future allocations of storage, but probably does not affect access to the presently stored data.
100020	check aborted error	Indicates that **dbcc checkstorage** stopped checking the table or index. To prevent multiple fault reports, the check operation on a single chain might be stopped without reporting this error. When an object contains several page chains, failure of the check operation for one chain does not prevent the continuation of the check operation on the other chains unless a fault threshold is exceeded.
100021	chain end error	Indicates that the end of the chain is corrupted. As a soft fault, it might indicate only that the chain was extended or truncated by more than a few pages during the **dbcc checkstorage** operation.
100022	chain start error	Indicates that the start of a chain is corrupted or is not at the expected location. If this is a persistent fault, access to data stored in the object is probably affected.
100023	used count error	Indicates an inconsistency between the count of the pages used that is recorded in the OAM page and the count of the pages used that is determined by examining the allocation pages.
100024	unused count error	Indicates an inconsistency between the count of the pages reserved but unused that is recorded in the OAM page and the count of the pages reserved but unused that is determined by examining the allocation pages.

Table 19-7: Contents of the dbcc_types table (continued)

type_code	*type_name*	Description
100025	row count error	Indicates an inconsistency between the row count recorded in the OAM page and the row count determined by **dbcc checkstorage**.
100026	serialloc error	Indicates a violation of the serial allocation rules applied to log allocations.
100027	text root error	Indicates a violation of the format of the root page of a *text* or *image* index. This check is similar to the root page checks performed by **dbcc textalloc**.
100028	page misplaced	Indicates that pages of this object were not found where they were expected to be from examination of the system tables. This usually indicates that **sp_placeobject** was used sometime in the past. In the *dbcc_counters* table, all misplaced pages are counted together, rather than being reported by device and partition.
100029	page header error	Indicates an internal inconsistency in the page's header other than the kind described by the other type codes. The severity of this error depends on the type of page and the inconsistency found.
100030	page format error	Indicates an internal inconsistency in the page's body other than the kind described by the other type codes. The severity of this error depends on the type of page and the inconsistency found.
100031	page not allocated	Indicates that **dbcc checkstorage** reached an unallocated page by following a page chain. This condition might affect access to data stored in this object.
100032	page linkage error	Indicates that **dbcc checkstorage** detected a fault with either the next or previous linkage of an interior page of a chain. If this is a persistent fault, access to data stored in the object is probably affected.

dbccdb Workspaces

Workspaces are special tables in *dbccdb* that store intermediate results of the **dbcc checkstorage** operation. Workspaces differ from worktables in that they:

- Are preallocated contiguously to improve I/O performance
- Are persistent
- Do not reside in the *tempdb* database

When you create *dbccdb*, two workspaces, *scan* and *text*, are created automatically. They are preallocated as follows:

- **Scan workspace** – contains a row for each page of the target database. The allocation is approximately 1 percent of the database size. Each row consists of a single *binary* (18) column.
- **Text workspace** – contains a row for each table in the target database that contains text or image columns. The size of this table depends on the design of the target database, but it is usually significantly smaller than the *scan* workspace. Each row consists of a single *binary* (22) column.

If either allocation is larger than needed by **dbcc checkstorage**, the operation uses only what it requires. The allocation does not change. If the *text* workspace allocation is too small, **dbcc checkstorage** reports this, recommends a new size, and continues checking; however, not all text chains are checked. If the *scan* workspace allocation is too small, the **dbcc checkstorage** operation fails immediately.

You must have at least one *scan* and one *text* workspace, but you may create as many as you need. While in use, the workspaces are locked so that only one **dbcc checkstorage** operation can use them at any given time. You can execute concurrent **dbcc checkstorage** operations by supplying each one with a separate *scan* and *text* workspace.

For more information on creating workspaces, see "Planning Resources" on page 18-28, and **sp_dbcc_createws** in the *Adaptive Server Reference Manual*.

Ideally, workspaces should be accessed only through **dbcc checkstorage**, but this is not a requirement. **dbcc checkstorage** exclusively locks the workspaces it uses, and the content of the workspaces is regenerated with each execution of **dbcc checkstorage**. The workspaces do not contain any secure data.

➤ ***Note***

While the contents of the workspaces are accessible through SQL, no interpretation of the binary values is available. Access through SQL might return data from different **dbcc** checks mixed together. The presence of a row in these tables does not ensure that it contains valid data. **dbcc** tracks valid rows only during execution. That information is lost when the operation completes.

Most of the update activity in *dbccdb* is performed in the *text* and *scan* workspaces. The workspaces are preallocated, and only one **dbcc**

checkstorage operation can use the workspaces at any given time, so the workspaces are less susceptible to corruption than most user tables. Corruption in a workspace can cause the **dbcc checkstorage** operation to fail or behave erratically. If this happens, drop and re-create the corrupt workspace.

Checks of databases using different workspaces can proceed simultaneously, but the performance of each operation might be degraded as it competes for disk throughput.

dbccdb Log

The results of each **dbcc checkstorage** operation are recorded in the *dbccdb* log. Updates to the *text* and *scan* workspaces are not recorded there.

The *dbccdb* log must be sized to handle updates to the tables. The log requirement is related to the number of tables and indexes in the target database. It is not related to the target database size.

To minimize the log requirement and the recovery time, use the **truncate log on checkpoint** option with *dbccdb*.

Backup and Recovery

20 Developing a Backup and Recovery Plan

Adaptive Server has **automatic recovery** procedures that protect you from power outages and computer failures. To protect yourself against media failure, you must make regular and frequent backups of your databases.

This chapter is the first of a four-chapter unit on **backup** and **recovery**. It provides information to help you develop a backup and recovery plan. It includes the following topics, which provide an overview of Adaptive Server's backup and recovery processes:

- Keeping Track of Database Changes 20-2
- Synchronizing a Database and Its Log: Checkpoints 20-3
- Automatic Recovery After a System Failure or Shutdown 20-6
- Using the Dump and Load Commands 20-17
- Designating Responsibility for Backups 20-25
- Using the Backup Server for Backup and Recovery 20-25
- Starting and Stopping Backup Server 20-30
- Configuring Your Server for Remote Access 20-30

This chapter also discusses the backup and recovery issues that you should address before you begin using your system for production, including:

- Choosing Backup Media 20-31
- Creating Logical Device Names for Local Dump Devices 20-32
- Scheduling Backups of User Databases 20-33
- Scheduling Backups of master 20-35
- Scheduling Backups of the model Database 20-37
- Scheduling Backups of the sybsystemprocs Database 20-38
- Gathering Backup Statistics 20-39

Table 20-1 lists additional sources of information about backup and recovery:

Table 20-1: Further information about backup and recovery

For More Information About	See
dump, **load**, **online database**, and **sp_volchanged** syntax	Chapter 21, "Backing Up and Restoring User Databases"
Backing up and restoring the system databases	Chapter 22, "Restoring the System Databases"
Using thresholds to automate backups	Chapter 23, "Managing Free Space with Thresholds"

Keeping Track of Database Changes

Adaptive Server uses transactions to keep track of all database changes. Transactions are Adaptive Server's units of work. A transaction consists of one or more Transact-SQL statements that succeed—or fail—as a unit.

Each SQL statement that modifies data is considered a **transaction**. Users can also define transactions by enclosing a series of statements within a **begin transaction...end transaction** block. For more information about transactions, see Chapter 18, "Transactions: Maintaining Data Consistency and Recovery," in the *Transact-SQL User's Guide*.

Each database has its own **transaction log**, the system table *syslogs*. The transaction log automatically records every transaction issued by each user of the database. You cannot turn off transaction logging.

The transaction log is a **write-ahead log**. When a user issues a statement that will modify the database, Adaptive Server writes the changes to the log. After all changes for a statement have been recorded in the log, they are written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page. At that time, it is written to disk.

If any statement in a transaction fails to complete, Adaptive Server reverses all changes made by the transaction. Adaptive Server writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction.

Getting Information About the Transaction Log

The transaction log contains enough information about each transaction to ensure that it can be recovered. Use the **dump transaction** command to copy the information it contains to tape or disk. Use **sp_spaceused syslogs** to check the size of the log, or **sp_helpsegment logsegment** to check the space available for log growth.

◆ ***WARNING!***

Never use insert, update, or delete commands to modify *syslogs*.

Synchronizing a Database and Its Log: Checkpoints

A checkpoint writes all dirty pages—pages that have been modified in memory, but not on disk, since the last checkpoint—to the database device. Adaptive Server's automatic **checkpoint** mechanism guarantees that data pages changed by completed transactions are regularly written from the cache in memory to the database device. Synchronizing the database and its transaction log shortens the time it takes to recover the database after a system crash. Figure 11-1 on page 11-24 illustrates the checkpoint process.

Setting the Recovery Interval

Typically, the automatic recovery process takes from a few seconds to a few minutes per database. The time varies, depending on the size of the database, the size of the transaction log, and the number and size of the transactions that must be committed or rolled back.

Use the **recovery interval in minutes** configuration parameter to specify, in minutes, the maximum permissible recovery time. Adaptive Server runs automatic checkpoints often enough to recover the database within that period of time.

Use **sp_configure** for a report on the current recovery interval:

```
sp_configure "recovery interval in minutes"
```

The default value, 5, allows recovery within 5 minutes per database. Use **sp_configure "recovery interval in minutes"** to change the recovery interval. For example, to set the recovery interval to 3 minutes, use this command:

```
sp_configure "recovery interval in minutes", 3
```

➤ ***Note***

The recovery interval has no effect on long-running, minimally logged transactions (such as **create index**) that are active at the time Adaptive Server fails. It may take as much time to reverse these transactions as it took to run them. To avoid lengthy delays, dump each database immediately after creating an index on one of its tables.

Automatic Checkpoint Procedure

Approximately once each minute, the checkpoint task "wakes up" and checks each database on the server to see how many records have been added to the transaction log since the last checkpoint. If the server estimates that the time required to recover these transactions is greater than the database's recovery interval, Adaptive Server issues a checkpoint.

The modified pages are written from cache onto the database devices, and the checkpoint event is recorded in the transaction log. Then, the checkpoint task "sleeps" for another minute.

To see the checkpoint task, execute **sp_who**. The checkpoint task is usually displayed as "CHECKPOINT SLEEP" in the "cmd" column:

```
spid  status     loginame   hostname blk dbname  cmd
----- ---------- ---------- -------- --- ------- ----------------
    1 running    sa         mars     0   master  SELECT
    2 sleeping   NULL                 0   master  NETWORK HANDLER
    3 sleeping   NULL                 0   master  MIRROR HANDLER
    4 sleeping   NULL                 0   master  HOUSEKEEPER
    5 sleeping   NULL                 0   master  CHECKPOINT SLEEP
```

Checkpoint After User Database Upgrade

Adaptive Server inserts a checkpoint record immediately after upgrading a user database. Adaptive Server uses this record to ensure that a **dump database** occurs before a **dump transaction** occurs on the upgraded database.

Truncating the Log After Automatic Checkpoints

System Administrators can use the **trunc log on chkpt** database option to truncate the transaction log when Adaptive Server performs an automatic checkpoint. If 50 or more rows are in the transaction log

when an automatic checkpoint occurs, Adaptive Server truncates the log.

To set the **trunc log on chkpt** database option, execute this command from the *master* database:

```
sp_dboption database_name, "trunc log on chkpt",
true
```

This option is not suitable for production environments because it does not make a copy of the transaction log before truncating it. Use **trunc log on chkpt** only for:

- Databases whose transaction logs cannot be backed up because they are not on a separate segment
- Test databases for which current backups are not important

➤ ***Note***

If you leave the **trunc log on chkpt** option set to **off** (the default condition), the transaction log continues to grow until you truncate it with the **dump transaction** command.

To protect your log from running out of space, you should design your last-chance threshold procedure to dump the transaction log. For more information about threshold procedures, see Chapter 23, "Managing Free Space with Thresholds."

Free Checkpoints

When Adaptive Server has no user tasks to process, a housekeeper task automatically begins writing dirty buffers to disk. If the housekeeper task is able to flush all active buffer pools in all configured caches, it wakes up the checkpoint task. The checkpoint task determines whether it needs to perform a checkpoint on the database.

Checkpoints that occur as a result of the housekeeper task are known as **free checkpoints**. They do not involve writing many dirty pages to the database device, since the housekeeper task has already done this work. They may result in a shorter recovery time for the database.

For information about tuning the housekeeper task, see Chapter 21, "How Adaptive Server Uses Engines and CPUs," in the *Performance and Tuning Guide*.

Manually Requesting a Checkpoint

Database Owners can issue the **checkpoint** command to force all modified pages in memory to be written to disk. Manual checkpoints do not truncate the log, even if the **trunc log on chkpt** option of **sp_dboption** is turned on.

Use the **checkpoint** command:

- As a precautionary measure in special circumstances—for example, just before a planned **shutdown with nowait** so that Adaptive Server's recovery mechanisms will occur within the recovery interval. (An ordinary **shutdown** performs a checkpoint.)
- To cause a change in database options to take effect after executing the **sp_dboption** system procedure. (After you run **sp_dboption**, an informational message reminds you to run **checkpoint**.)

Automatic Recovery After a System Failure or Shutdown

Each time you restart Adaptive Server—for example, after a power failure, an operating system failure, or the use of the **shutdown** command—it automatically performs a set of recovery procedures on each database.

The recovery mechanism compares each database to its transaction log. If the log record for a particular change is more recent than the data page, the recovery mechanism reapplies the change from the transaction log. If a transaction was ongoing at the time of the failure, the recovery mechanism reverses all changes that were made by the transaction. This mechanism ensures that the entire transaction succeeds or fails as a unit.

When Adaptive Server is booted, it performs database recovery in this order:

1. Recovers *master*
2. Recovers *model*
3. Creates *tempdb* (by copying *model*)
4. Recovers *sybsystemdb*
5. Recovers *sybsecurity*
6. Recovers *sybsystemprocs*
7. Recovers user databases, in order by *sysdatabases.dbid*

Users can log into Adaptive Server as soon as the system databases have been recovered, but they cannot access other databases until they have been recovered.

Determining Whether Messages Are Displayed During Recovery

The configuration variable **print recovery information** determines whether Adaptive Server displays detailed messages about each transaction on the console screen during recovery. By default, these messages are not displayed. To display messages, use this command:

```
sp_configure "print recovery information", 1
```

Fault Isolation During Recovery

The recovery procedures, known simply as "recovery", rebuild the server's databases from the transaction logs. Four situations cause recovery to run:

- Adaptive Server start-up
- Use of the **load database** command
- Use of the **load transaction** command
- Use of the **dbcc dbrecover** command.

The recovery isolation mode setting controls how recovery behaves when it detects corrupt data while reversing or reapplying a transaction in a database.

If the recovery isolation mode is "database", and recovery detects data corruption, it marks the entire database suspect. When a database is marked suspect, it is not accessible to users. This is the default behavior.

If the recovery isolation mode is "page", and the corruption can be identified with specific pages, recovery marks only the corrupt pages suspect. The suspect pages are not accessible to users, but the rest of the database is accessible, unless the mode is specified as **read_only,** in which case the entire database comes online, but only for reading, not for writing.

Recovery fault isolation provides the ability to:

- Configure whether an entire database or just the suspect pages become inaccessible when recovery detects corruption

- Configure whether an entire database with suspect pages comes online in **read_only** mode or whether the online pages are accessible for modification
- List databases that have suspect pages
- List the suspect pages in a specified database by page ID, index ID, and object name
- Bring suspect pages online for the System Administrator while they are being repaired
- Bring suspect pages online for all database users after they have been repaired

The ability to isolate only the suspect pages while bringing the rest of the database online provides a greater degree of flexibility in dealing with data corruption. You can diagnose problems, and sometimes correct them, while most of the database is accessible to users. You can assess the extent of the damage and schedule emergency repairs or reload for a convenient time.

Recovery fault isolation applies only to user databases. Recovery always takes a system database entirely offline if it has any corrupt pages. A system database cannot be recovered until all of its corrupt pages have been repaired or removed.

Persistence of Offline Pages

Suspect pages that have been taken offline remain offline when you reboot the server. Information about offline pages is stored in *master.dbo.sysattributes*.

The **drop database** and **load database** commands clear the database's entries for suspect pages from *master.dbo.sysattributes*.

Configuring Recovery Fault Isolation

When Adaptive Server is installed, the default behavior is for recovery to mark a database as suspect and take the entire database offline if it detects any corrupt pages. In other words, the default recovery isolation mode for a database is "database."

Isolating Suspect Pages

To isolate the suspect pages so that only they are taken offline, while the rest of the database remains accessible to users, use the

sp_setsuspect_granularity system procedure to set the recovery isolation mode to "page." This mode will be in effect the next time that recovery is performed in the database.

The syntax for the **sp_setsuspect_granularity** system procedure is:

```
sp_setsuspect_granularity
   [dbname [,{"database" | "page"} [, "read_only"]]]
```

With the *dbname* and either **database** or **page** as the second argument, **sp_setsuspect_granularity** sets the recovery isolation mode.

Without the **database** or **page** argument, **sp_setsuspect_granularity** displays the current and configured recovery isolation mode settings for the specified database. Without any arguments, it displays those settings for the current database.

If corruption cannot be isolated to a specific page, recovery marks the entire database as suspect, even if you set the recovery isolation mode to "page." For example, a corrupt transaction log or the unavailability of a global resource causes this to occur.

When recovery marks specific pages suspect, the default behavior is for the database to come online (that is, to be available) with the suspect pages offline (unavailable) and therefore inaccessible but for the rest of the database to be accessible for reading and writing. However, if you specify the **read_only** option to **sp_setsuspect_granularity,** and recovery marks any pages suspect, the entire database comes online in **read_only** mode and cannot be modified. If the **read_only** option is normally preferred, but in certain cases you are comfortable allowing users to modify the non-suspect pages, you can make the online portion of the database writeable with **sp_dboption**:

```
sp_dboption pubs2, "read only", false
```

In this case, the suspect pages remain offline until they are repaired or forced online, as described in "Bringing Offline Pages Online" on page 20-11.

Raising the Number of Suspect Pages Allowed

The suspect escalation threshold is the number of suspect pages at which recovery marks an entire database suspect, even if the recovery isolation mode is "page." By default, it is set to 20 pages in a single database. You can use the **sp_setsuspect_threshold** system procedure to raise or lower the suspect escalation threshold.

The syntax for **sp_setsuspect_threshold** is:

```
sp_setsuspect_threshold [dbname [,threshold]]
```

With the *dbname* and *threshold* arguments, **sp_setsuspect_threshold** displays the current and configured suspect escalation threshold settings for the specified database. Without any arguments, it displays these settings for the current database.

You configure recovery fault isolation and the suspect escalation threshold at the database level.

You cannot execute **sp_setsuspect_granularity** or **sp_setsuspect_threshold** inside a transaction.

You must have the **sa_role** and be in the *master* database to set values with the **sp_setsuspect_granularity** and **sp_setsuspect_threshold** system procedures. Any user can execute these procedures with only the name of the database as an argument to display the values configured for that database, as illustrated below:

```
sp_setsuspect_granularity pubs2
DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  -----------------  ------------------  -----------
pubs2    page               page                read/write
sp_setsuspect_threshold pubs2
DB Name       Cur. Suspect threshold   Cfg. Suspect threshold
------------- ----------------------- ----------------------
pubs2         20                       30
```

This example shows that the recovery isolation mode for the *pubs2* database was "page" and the escalation threshold was 20 the last time recovery ran on this database (the current suspect threshold values). The next time recovery runs on this database, the recovery isolation mode will be "page" and the escalation threshold will be 30 (the configured values).

With no arguments, **sp_setsuspect_granularity** and **sp_setsuspect_threshold** display the current and configured settings for the current database, if it is a user database.

Getting Information About Offline Databases and Pages

To see which databases have offline pages, use the **sp_listsuspect_db** system procedure. The syntax is:

```
sp_listsuspect_db
```

The following example displays general information about the suspect pages:

```
sp_listsuspect_db
The database 'dbt1' has 3 suspect pages belonging to 2 objects.
(return status = 0)
```

To get detailed information about the individual offline pages, use the **sp_listsuspect_page** system procedure. The syntax is:

```
sp_listsuspect_page [dbname]
```

If the *dbname* is not specified, it defaults to the current database. The following example shows the detailed page-level output of **sp_listsuspect_page** in the *dbt1* database.

```
sp_listsuspect_page dbt1
DBName   Pageid       Object    Index   Access
-------- ------------ --------- ------- -------------------
dbt1     384          tab1      0       BLOCK_ALL
dbt1     390          tab1      0       BLOCK_ALL
dbt1     416          tab1      1       SA_ONLY

(3 rows affected, return status = 0)
```

If the value in the "Access" column is SA_ONLY, the suspect page is 1, the suspect page is accessible to users with the **sa_role**. If it is BLOCK_ALL, no one can access the page.

Any user can run **sp_listsuspect_db** and **sp_listsuspect_page** from any database.

Bringing Offline Pages Online

To make all the offline pages in a database accessible, use the **sp_forceonline_db** system procedure. The syntax is:

```
sp_forceonline_db dbname,
   {"sa_on" | "sa_off" | "all_users"}
```

To make an individual offline page accessible, use **sp_forceonline_page**. The syntax is:

```
sp_forceonline_page dbname, pgid
   {"sa_on" | "sa_off" | "all_users"}
```

With both of these procedures, you specify the type of access as **"sa_on"**, **"sa_off"**, or **"all_users"**.

- Specify **"sa_on"** to make the suspect page or database accessible only to users with the **sa_role**. This is useful for repairing the suspect pages and testing the repairs while the database is up and running, without allowing normal users access to the suspect

pages. You can also use it to perform a **dump database** or a **dump transaction with no_log** on a database with suspect pages, which would be prohibited if the pages were offline.

- Specify **"sa_off"** to block access to all users, including System Administrators. This reverses a previous **sp_forceonline_db** or **sp_forceonline_page** with **"sa_on".**
- Specify **"all_users"** to bring offline pages online for all users after the pages have been repaired.

  Unlike bringing suspect pages online for with **"sa_on"** and then making them offline again with **"sa_off"**, when you use **sp_forceonline_page** or **sp_forceonline_db** to bring pages online for "all users", this action cannot be reversed. There is no way to make the online pages offline again.

◆ ***WARNING!***

Adaptive Server does not perform any checks on pages being brought online. It is your responsibility to ensure that pages being brought online for all users have been repaired.

You cannot execute **sp_forceonline_db** or **sp_forceonline_page** inside a transaction.

You must have the **sa_role** and be in the *master* database to execute **sp_forceonline_db** and **sp_forceonline_page.**

Side Effects of Offline Pages

The following restrictions apply to databases with offline pages:

- Transactions that need offline data, either directly or indirectly (for example, because of referential integrity constraints), fail and generate a message.
- You cannot use **dump database** when any part of the database is offline.

  A System Administrator can force the offline pages online using **sp_forceonline_db** with **"sa_on"**, dump the database, and then use **sp_forceonline_db** with **"sa_off"** after the dump completes.
- You cannot use **dump transaction with no_log** or **dump transaction with truncate_only** if any part of a database is offline.

A System Administrator can force the offline pages online using **sp_forceonline_db** with **"sa_on"**, dump the transaction log using **with no_log**, and then use **sp_forceonline_db** with **"sa_off"** after the dump completes.

- If you want to drop a table or index containing offline pages, you must use a transaction in the *master* database. Otherwise, the drop will fail, because it needs to delete entries for the suspect pages from *master.dbo.sysattributes*. The following example shows how to drop an index that contains suspect pages. It drops the object and deletes information about its offline pages from *master.dbo.sysattributes*.

  To drop an index named *authors_au_id_ind*, which contains suspect pages, from the *pubs2* database, drop the index inside a *master* database transaction as follows:

```
use master
go
sp_dboption pubs2, "ddl in tran", true
go
use pubs2
go
checkpoint
go
begin transaction
drop index authors.au_id_ind
commit
go
use master
go
sp_dboption pubs2, "ddl in tran", false
go
use pubs2
go
checkpoint
go
```

  See "Using the sp_dboption Procedure" on page 16-1 for information on **sp_dboption**.

Recovery Strategies Using Recovery Fault Isolation

There are two major strategies for returning a database with suspect pages to a consistent state while users are accessing it: reload and repair.

Both strategies require:

- A clean database dump
- A series of reliable transaction log dumps up to the point at which the database is recovered with suspect pages
- A transaction log dump to a device immediately after the database is recovered to capture changes to the offline pages
- Continuous transaction log dumps to devices while users work in the partially offline database

Reload Strategy

The reload strategy involves restoring a clean database from backups. When convenient, load the most recent clean database dump, and apply the transaction logs to restore the database.

load database clears the suspect page information from the *master.dbo.sysdatabases* and *master.dbo.sysattributes* system tables.

When the restored database is online, dump the database immediately.

Figure 20-1 illustrates the strategy used to reload databases.

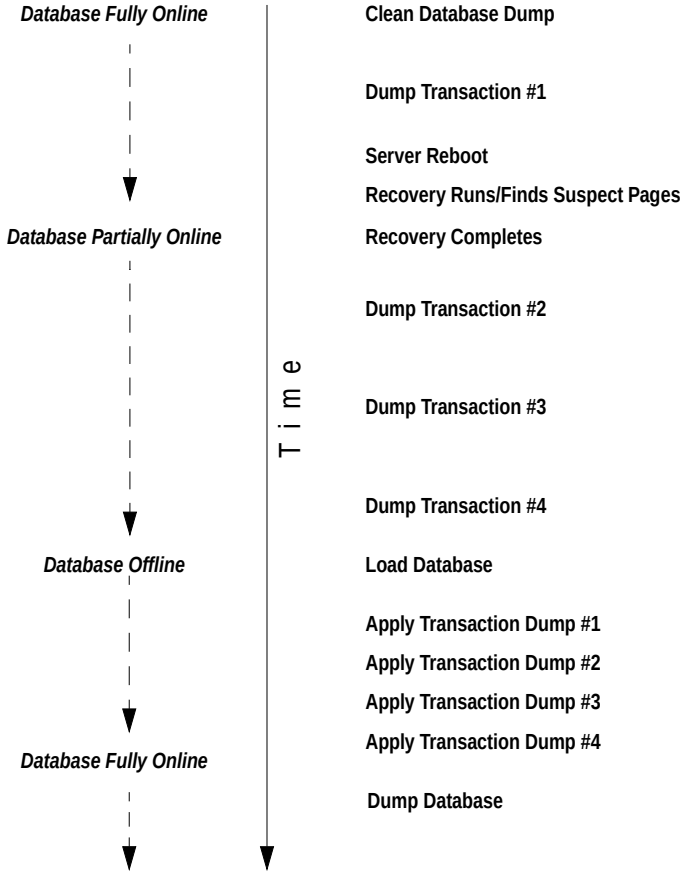


Figure 20-1: Reload strategy

Repair Strategy

The repair strategy involves repairing the corrupt pages while the database is partially offline. You diagnose and repair problems using known methods, including **dbcc** commands, running queries with known results against the suspect pages, which have been forced online for System Administrator use only, and calling Sybase Technical Support, if necessary. Repairing damage can also include dropping and re-creating objects that contain suspect pages.

You can either use **sp_forceonline_page** to bring offline pages online individually, as they are repaired, or wait until all the offline pages are repaired and bring them online all at once with **sp_forceonline_db**.

An advantage of the repair strategy is that it does not require taking the entire database offline. Figure 20-2 illustrates the strategy used to repair corrupt pages.

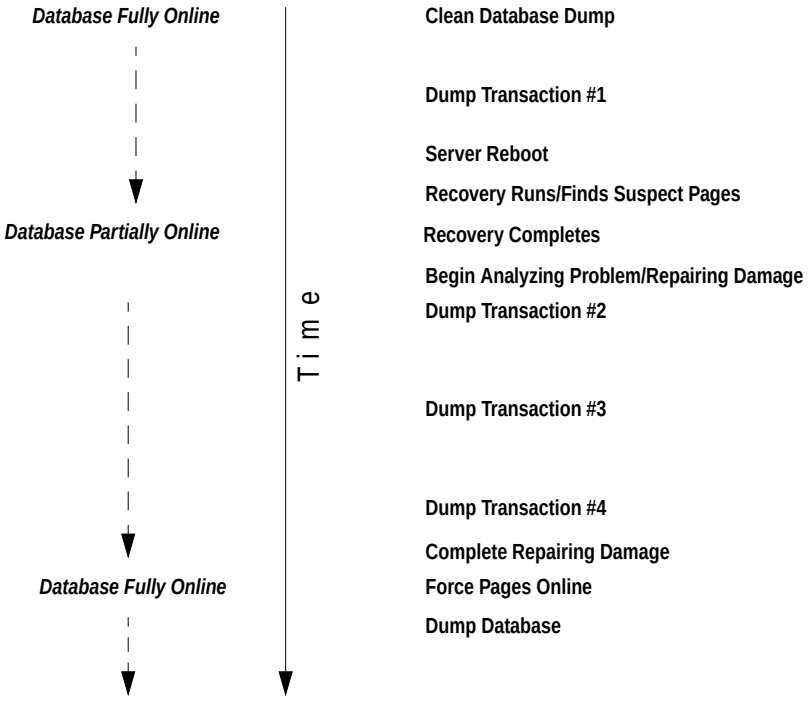


Figure 20-2: Repair strategy

Assessing the Extent of Corruption

You can sometimes use recovery fault isolation to assess the extent of corruption by forcing recovery to run and examining the number of pages marked suspect and the objects to which they belong.

For example, if users report problems in a particular database, set the recovery isolation mode to "page" and force recovery either by running **dbcc dbrecover** or by restarting Adaptive Server. When recovery completes, you can use **sp_listsuspect_db** or **sp_listsuspect_page**

to determine how many pages are suspect and which database objects are affected.

If the entire database is marked suspect and you receive the message:

```
Reached suspect threshold '%d' for database
'%.*s'. Increase suspect threshold using
sp_setsuspect_threshold.
```

use **sp_setsuspect_threshold** to raise the suspect escalation threshold and force recovery to run again. Each time you get this message, you can raise the threshold and run recovery until the database comes online. If you do not get this message, the corruption is not isolated to specific pages, in which case this strategy for determining the number of suspect pages will not work.

Using the Dump and Load Commands

In case of media failure, such as a disk crash, you can restore your databases if—and only if—you have been making regular backups of the databases and their transaction logs. Full recovery depends on the regular use of the **dump database** and **dump transaction** commands to back up databases and the **load database** and **load transaction** commands to restore them. These commands are described briefly below and more fully in Chapter 21, "Backing Up and Restoring User Databases," and Chapter 22, "Restoring the System Databases."

◆ ***WARNING!***

Never use operating system copy commands to copy a database device. Loading the copy into Adaptive Server causes massive database corruption.

Checking Database Consistency: *dbcc*

The dump commands can complete successfully even if your database is corrupt. Before you back up a database, use the **dbcc** commands to check its consistency. See Chapter 18, "Checking Database Consistency," for more information.

Making Routine Database Dumps: *dump database*

The **dump database** command makes a copy of the entire database, including both the data and the transaction log. **dump database** does **not** truncate the log.

dump database allows **dynamic dumps**. Users can continue to make changes to the database while the dump takes place. This feature makes it convenient to back up databases on a regular basis.

The **dump database** command executes in three phases. A progress message informs you when each phase completes. When the dump is finished, it reflects all changes that were made during its execution, except for those initiated during phase 3.

Making Routine Transaction Log Dumps: *dump transaction*

Use the **dump transaction** command (or its abbreviation, **dump tran**) to make routine backups of your transaction log. **dump transaction** is similar to the incremental backups provided by many operating systems. It copies the transaction log, providing a record of any database changes made since the last database or transaction log dump. Once **dump transaction** has copied the log, it truncates the inactive portion.

dump transaction takes less time and storage space than a full database backup, and it is usually run more often. Users can continue to make changes to the database while the dump is taking place. You can run **dump transaction** only if the database stores its log on a separate segment.

After a media failure, use the **with no_truncate** option of **dump transaction** to backup your transaction log. This provides a record of the transaction log up to the time of the failure.

Copying the Log After Device Failure: *dump tran with no_truncate*

When your data device fails and the database is inaccessible, use the **with no_truncate** option of the **dump transaction** command to get a current copy of the log. This option does not truncate the log. You can use it only if the transaction log is on a separate segment and the *master* database is accessible.

Restoring the Entire Database: *load database*

Use the **load database** command to load the backup created with **dump database**. You can load the dump into a pre-existing database or create a new database with the **for load** option. When you create a new database, allocate at least as much space as was allocated to the original database.

◆ ***WARNING!***

You cannot load a dump that was made on a different platform or generated on SQL Server prior to release 10.0. If the database you are loading includes tables that contain the primary keys for tables in other databases, you must load the dump into a database with the same database name as the one dumped.

The **load database** command sets the database status to "offline." The "offline" status means that it is not necessary to use the **no chkpt on recovery, dbo use only,** and **read only** options of **sp_dboption** before loading a database. However, no one can use a database while it is offline, thus preventing users from making changes to the database during the database load and subsequent transaction log loads. To make the database accessible to users, issue the **online database** command.

After the database is loaded, Adaptive Server may need to complete two more tasks:

- It must "zero" all unused pages, if the database being loaded into is larger than the dumped database.
- It must complete recovery, applying transaction log changes to the data.

Depending on the number of unallocated pages or the number of long transactions, this can take a few seconds or many hours for a very large database. Adaptive Server will issue messages telling you that it is "zero-ing" pages or that it has begun recovery. These messages are normally buffered; in order to see them, issue this command:

```
set flushmessage on
```

Applying Changes to the Database: *load transaction*

Once you have loaded the database, use the **load transaction** command (or its abbreviation, **load tran**) to load each transaction log dump **in the**

order in which it was made. This process reconstructs the database by reexecuting the changes recorded in the transaction log. If necessary, you can recover a database, by rolling it forward to a particular time in its transaction log, by specifying the **until_time** option of the **load transaction** command.

Users cannot make changes to the database between the **load database** and **load transaction** commands, due to the "offline" status set by **load database**.

You can load only the transaction log dumps that are at the same release level as the associated database.

When the entire sequence of transaction log dumps has been loaded, the database reflects all transactions that had committed at the time of the last transaction log dump.

Making the Database Available to Users: *online database*

When the load sequence completes, use the **online database** command to change the database status to "online," thus making it available to users. A database loaded by **load database** remains inaccessible until the **online database** command is issued.

Be sure you have loaded all required transaction logs before issuing the **online database** command.

The **online database** command also upgrades user databases, as described in the next section.

Moving a Database to Another Adaptive Server

You can use the **dump database** and **load database** commands to move a database from one Adaptive Server to another, as long as both Adaptive Servers run on the same hardware and software platform. However, you must take steps to ensure that the device allocations on the target Adaptive Server match those on the original. Otherwise, system-defined and user-defined segments in the new database will not match those in the original database.

To preserve device allocations when loading a database dump into a new Adaptive Server, use the same instructions as you would for recovering a user database from a failed device. See "Examining the Space Usage" on page 21-40 for more information.

Also, follow these general guidelines when moving system databases to different devices:

- Before moving the *master* database, always unmirror the master device. If you do not unmirror the device, Adaptive Server will try to use the old mirror device file when you start Adaptive Server with the new device.
- When moving the *master* database, use a new device that is the same size as the original to avoid allocation errors in *sysdevices*.
- To move the *sybsecurity* database, place the new database in single-user mode before loading the old data into it.

Upgrading a Single-User Database

You can upgrade SQL Server release 10.0 database and transaction log dumps for a single-user database to Adaptive Server release 11.5.x Beta.

➤ ***Note***

Upgrading a single-user database can occur only on databases in releases 10.0 or later.

The steps for upgrading a user database are the same as for a normal database load:

1. Use **load database** to load the most recent SQL Server release 10.0 database dump. **load database** sets the database status to "offline."
2. Use **load transaction** to load, **in order**, all SQL Server release 10.0 transaction logs generated after the last database dump. Be sure you have loaded all transaction logs before going to step 3.
3. Use **online database** to upgrade the database to Adaptive Server release 11.5.x Beta. When the upgrade completes, the database status is set to "online," which makes the database available for "public" use.
4. Use **dump database** on the newly upgraded database to create a dump that is consistent with Adaptive Server release 11.5.x Beta. A **dump database** must occur before a **dump transaction** command is permitted.

For additional information regarding the **load database**, **load transaction**, and **online database** commands in relation to upgrading a user database, see the *Adaptive Server Reference Manual*.

Using the Special *dump transaction* Options

In certain circumstances, the simple model described above does not apply. Table 20-2 describes when to use the special **with no_log** and **with truncate_only** options instead of the standard **dump transaction** command.

◆ ***WARNING!***

Use the special dump transaction commands only as indicated in Table 20-2. In particular, use dump transaction with no_log as a last resort and use it only once after dump transaction with no_truncate fails. The dump transaction with no_log command frees very little space in the transaction log. If you continue to load data after entering dump transaction with no_log, it is possible to fill the log completely, causing any further dump transaction commands to fail. Use the alter database command to allocate additional space to the database.

Table 20-2: When to use dump transaction with truncate_only or with no_log

When	Use
The log is on the same segment as the data	**dump transaction with truncate_only** to truncate the log **dump database** to copy the entire database, including the log
You are not concerned with the recovery of recent transactions (for example, in an early development environment)	**dump transaction with truncate_only** to truncate the log **dump database** to copy the entire database
Your usual method of dumping the transaction log (either the standard **dump transaction** command or **dump transaction with truncate_only**) fails because of insufficient log space	**dump transaction with no_log** to truncate the log without recording the event **dump database** immediately afterward to copy the entire database, including the log

Using the Special Load Options to Identify Dump Files

Use the **with headeronly** option to provide header information for a specified file or for the first file on a tape. Use the **with listonly** option to return information about all files on a tape. These options do not actually load databases or transaction logs on the tape.

➤ ***Note***

These options are mutually exclusive. If you specify both, **with listonly** prevails.

Restoring a Database from Backups

Figure 20-3 illustrates the process of restoring a database that is created at 4:30 p.m. on Monday and dumped immediately afterward. Full database dumps are made every night at 5:00 p.m. Transaction log dumps are made at 10:00 a.m., 12:00 p.m., 2:00 p.m., and 4:00 p.m. every day:

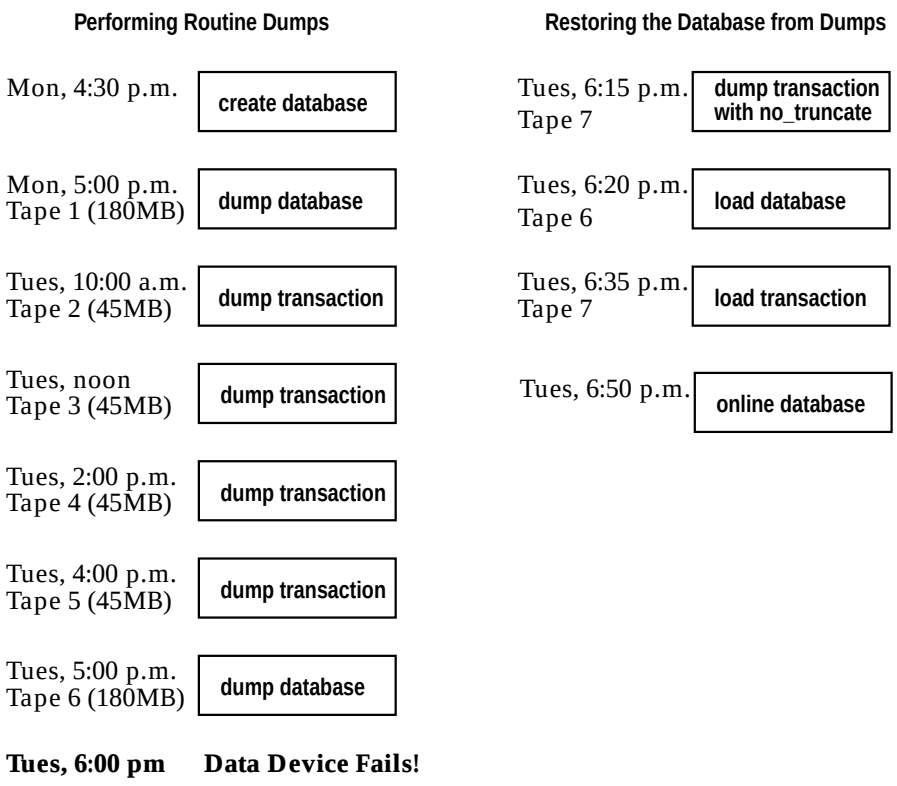


Figure 20-3: Restoring a database, a scenario

If the disk that stores the data fails on Tuesday at 6:00 p.m., follow these steps to restore the database:

1. Use **dump transaction with no_truncate** to get a current transaction log dump.
2. Use **load database** to load the most recent database dump, Tape 6. **load database** sets the database status to "offline."
3. Use **load transaction** to apply the most recent transaction log dump, Tape 7.
4. Use **online database** to set the database status to "online."

Figure 20-4 illustrates how to restore the database when the data device fails at 4:59 p.m. on Tuesday—just before the operator is scheduled to make the nightly database dump:

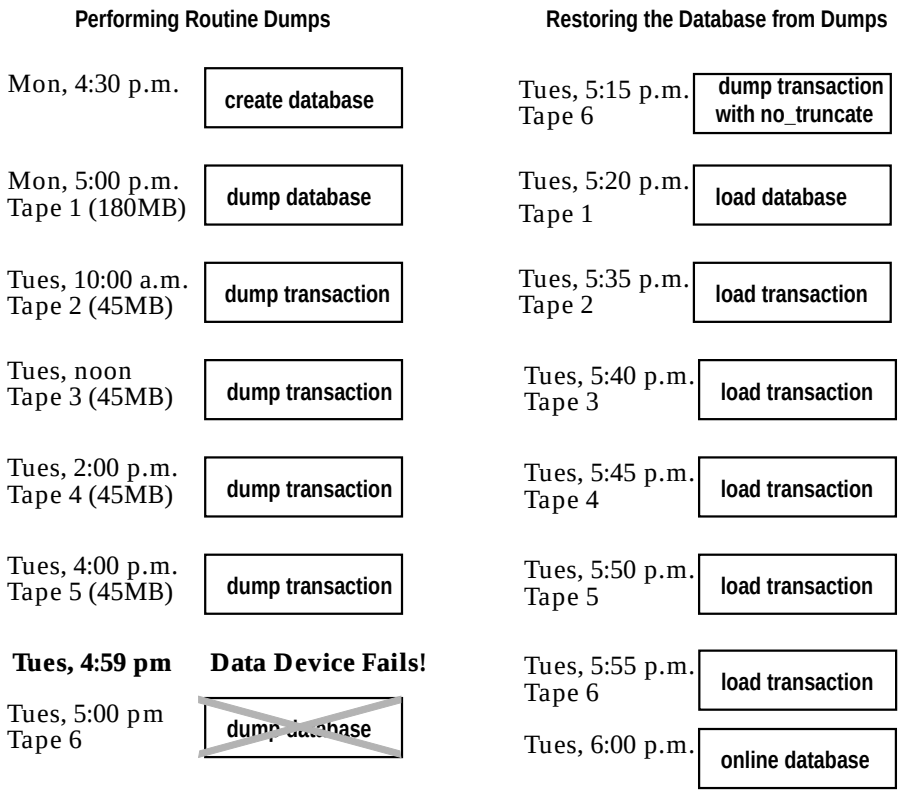


Figure 20-4: Restoring a database, a second scenario

Use the following steps to restore the database:

1. Use **dump transaction with no_truncate** to get a current transaction log dump on Tape 6 (the tape you would have used for the routine database dump).
2. Use **load database** to load the most recent database dump, Tape 1. **load database** sets the database status to "offline."
3. Use **load transaction** to load Tapes 2, 3, 4, and 5 and the most recent transaction log dump, Tape 6.
4. Use **online database** to set the database status to "online."

Designating Responsibility for Backups

Many organizations have an operator whose job is to perform all backup and recovery operations. Only a System Administrator, a Database Owner, or an Operator can execute the dump and load commands. The Database Owner can dump only his or her own database. The Operator and System Administrator can dump and load any database.

Any user can execute **sp_volchanged** to notify the Backup Server when a tape volume is changed. On OpenVMS systems, the operator responsible for changing tape volumes must have permission to execute the **REPLY** command.

Using the Backup Server for Backup and Recovery

Dumps and loads are performed by an Open Server program, Backup Server, running on the same machine as Adaptive Server. You can perform backups over the network, using a Backup Server on a remote computer and another on the local computer.

Backup Server:

- Creates and loads from "striped dumps." **Dump striping** allows you to use up to 32 backup devices in parallel. This splits the database into approximately equal portions and backs up each portion to a separate device.
- Creates and loads single dumps that span several tapes.
- Dumps and loads over the network to a Backup Server running on another machine.
- Dumps several databases or transaction logs onto a single tape.

- Loads a single file from a tape that contains many database or log dumps.
- Supports platform-specific tape handling options.
- Directs volume-handling requests to the session where the dump or load command was issued or to its operator console.
- Detects the physical characteristics of the dump devices to determine protocols, block sizes, and other characteristics.

Relationship Between Adaptive Server and Backup Servers

Figure 20-5 shows two users performing backup activities simultaneously on two databases:

- User1 is dumping database *db1* to a remote Backup Server.
- User2 is loading database *db2* from the local Backup Server.

Each user issues the appropriate dump or load command from a Adaptive Server session. Adaptive Server interprets the command and sends remote procedure calls (RPCs) to the Backup Server. The calls indicate which database pages to dump or load, which dump devices to use, and other options.

While the dumps and loads execute, Adaptive Server and Backup Server use RPCs to exchange instructions and status messages. Backup Server—not Adaptive Server—performs all data transfer for the dump and load commands.

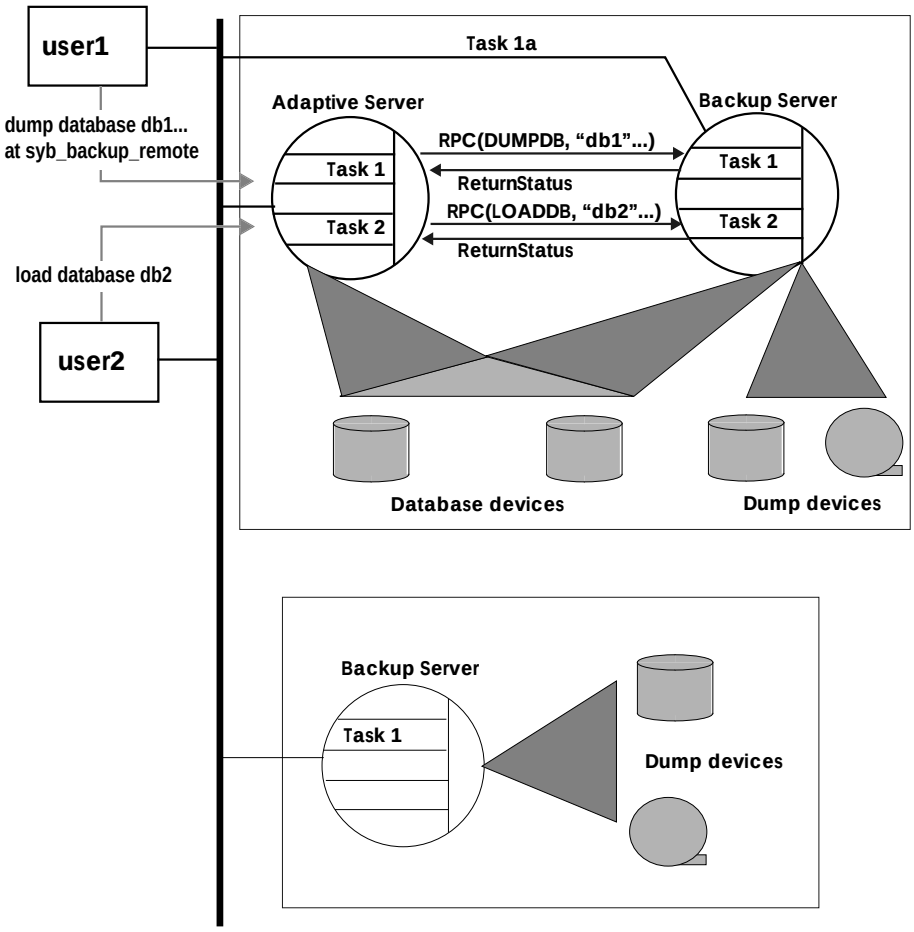


Figure 20-5: Adaptive Server and Backup Server with remote Backup Server

When the local Backup Server receives user1's dump instructions, it reads the specified pages from the database devices and sends them to the remote Backup Server. The remote Backup Server saves the data to offline media.

Simultaneously, the local Backup Server performs user2's load command by reading data from local dump devices and writing it to the database device.

Communicating with the Backup Server

To use the dump and load commands, an Adaptive Server must be able to communicate with its Backup Server. These are the requirements:

- The Backup Server must be running on the same machine as the Adaptive Server (or on the same cluster for OpenVMS).
- The Backup Server must be listed in the *master..sysservers* table. The Backup Server entry, SYB_BACKUP, is created in *sysservers* when you install Adaptive Server. Use **sp_helpserver** to see this information.
- The Backup Server must be listed in the interfaces file. The entry for the local Backup Server is created when you install Adaptive Server. The name of the Backup Server listed in the interfaces file must match the column *srvnet* name for the SYB_BACKUP entry in *master..sysservers*. If you have installed a remote Backup Server on another machine, create the interfaces file on a file system that is shared by both machines, or copy the entry to your local interfaces file. The name of the remote Backup Server must be the same in both interfaces files.
- The user who starts the Backup Server process must have write permission for the dump devices. The "sybase" user, who usually starts Adaptive Server and Backup Server, can read from and write to the database devices.
- Adaptive Server must be configured for remote access. By default, Adaptive Server is installed with remote access enabled. See "Configuring Your Server for Remote Access" on page 20-30 for more information.

Mounting a New Volume

During the backup and restore process, it may be necessary to change tape volumes. If the Backup Server detects a problem with the currently mounted volume, it requests a volume change by sending messages to either the client or its operator console. After mounting another volume, the operator notifies the Backup Server by executing the **sp_volchanged** system procedure on Adaptive Server.

On UNIX systems, the Backup Server requests a volume change when the tape capacity has been reached. The operator mounts another tape and then executes **sp_volchanged** (see Table 20-3.)

On OpenVMS systems, the operating system requests a volume change when it detects the end of a volume or when the specified drive is offline. The operator uses the **REPLY** command to reply to these messages.

Table 20-3: Changing tape volumes on a UNIX system

Sequence	Operator Using *isql*	Adaptive Server	Backup Server
1	Issues the **dump database** command		
2		Sends dump request to Backup Server	
3			Receives dump request message from Adaptive Server Sends message for tape mounting to operator Waits for operator's reply
4	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		
5			Checks tapes If tapes are okay, begins dump When tape is full, sends volume change request to operator
6	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		
7			Continues dump When dump is complete, sends messages to operator and Adaptive Server

Table 20-3: Changing tape volumes on a UNIX system (continued)

Sequence	Operator Using *isql*	Adaptive Server	Backup Server
8	Receives message that dump is complete	Receives message that dump is complete	
	Removes and labels tapes	Releases locks	
		Completes the **dump database** command	

Starting and Stopping Backup Server

Most UNIX systems use the **startserver** utility to start Backup Server on the same machine as Adaptive Server. On Windows NT, you can start Backup Server from Sybase Central. See the configuration documentation for your platform for information about starting Backup Server on your system.

Use the **shutdown** command to shut down a Backup Server. See Chapter 4, "Diagnosing System Problems," and the *Adaptive Server Reference Manual* for information about this command.

Configuring Your Server for Remote Access

The **remote access** configuration parameter is set to 1 when you install Adaptive Server. This allows Adaptive Server to execute remote procedure calls to the Backup Server.

For security reasons, you may want to disable remote access except during times when dumps and loads are taking place. Use the following command to disable remote access:

```
sp_configure "allow remote access", 0
```

Before you perform a dump or load, use the following command to reenable remote access:

```
sp_configure "allow remote access", 1
```

The **allow remote access** configuration parameter is dynamic and does not require a restart of Adaptive Server to take effect. Only a System Security Officer can set the **allow remote access** parameter.

Choosing Backup Media

Tapes are preferred as dump devices, since they permit a library of database and transaction log dumps to be kept offline. Large databases can span multiple tape volumes. On UNIX systems, the Backup Server requires nonrewinding tape devices for all dumps and loads.

For a list of supported dump devices, see the the configuration documentation for your platform.

Protecting Backup Tapes from Being Overwritten

The **tape retention in days** configuration parameter determines how many days' backup tapes are protected from being overwritten. When you install Adaptive Server, **tape retention in days** has a value of 0. This means that backup tapes can be overwritten immediately.

Use **sp_configure** to change the **tape retention in days** value. The new value takes effect the next time you restart Adaptive Server:

```
sp_configure "tape retention in days", 14
```

Both the **dump database** and **dump transaction** commands provide a **retaindays** option that overrides the **tape retention in days** value for that dump.

Dumping to Files or Disks

In general, dumping to a file or disk is not recommended. If the disk or computer containing that file crashes, there may be no way to recover the dumps. On UNIX and PC systems, the entire master database dump must fit into a single volume. On these systems, dumping to a file or disk is your only option if the *master* database is too large to fit on a single tape volume, unless you have a second Adaptive Server that can issue **sp_volchanged** requests.

Dumps to a file or disk can be copied to tape for offline storage, but these tapes must be copied back to an online file before they can be read by Adaptive Server. A dump that is made to a disk file and then copied to tape cannot be read directly from tape by Backup Server.

Creating Logical Device Names for Local Dump Devices

If you are dumping to or loading from local devices (that is, if you are not performing backups over a network to a remote Backup Server), you can specify dump devices either by providing their physical locations or by specifying their logical device names. In the latter case, you may want to create logical dump device names in the *sysdevices* system table of the *master* database.

➤ ***Note***

If you are dumping to or loading from a remote Backup Server, you must specify the absolute path name of the dump device. You cannot use a logical device name.

The *sysdevices* table stores information about each database and backup device, including its *physical_name* (the actual operating system device or file name) and its *device_name* (or logical name, known only within Adaptive Server). On most platforms, Adaptive Server has one or two aliases for tape devices installed in *sysdevices*. The physical names for these devices are common disk drive names for the platform; the logical names are *tapedump1* and *tapedump2*.

When you create backup scripts and threshold procedures, use logical names, rather than physical device names, whenever possible. Scripts and procedures that refer to actual device names must be modified each time a backup device is replaced. If your scripts and procedures refer to logical device names, you can simply drop the *sysdevices* entry for the failed device and create a new entry that associates the logical name with a different physical device.

Listing the Current Device Names

To list the backup devices for your system, run the following query:

```
select * from master..sysdevices
  where status = 16 or status = 24
```

To list both the physical and logical names for database and backup devices, use the **sp_helpdevice** system procedure:

```
sp_helpdevice tapedump1
```

```
device_name physical_name
 description
 status cntrltype device_number low      high
 ------ --------- ------------- -------- -------
tapedump1 /dev/nrmt4
tape, 625 MB, dump device
   16          3             0        0   20000
```

Adding a Backup Device

Use the system procedure **sp_addumpdevice** to add a backup device:

```
sp_addumpdevice{ "tape" | "disk"} , logicalname,
   physicalname, tapesize
```

The *physicalname* can be either an absolute path name or a relative path name. During dumps and loads, the Backup Server resolves relative path names by looking in Adaptive Server's current working directory.

The *tapesize* is the capacity of the tape in megabytes. OpenVMS systems ignore the *tapesize* parameter if it is specified. Other platforms require this parameter for tape devices but ignore it for disk devices. The Backup Server uses the *tapesize* parameter if the dump command does not specify a tape capacity.

The *tapesize* must be at least 1MB and should be slightly below the capacity rated for the device.

Redefining a Logical Device Name

To use an existing logical device name for a different physical device, drop the device with **sp_dropdevice** and then add it with **sp_addumpdevice**. For example:

```
sp_dropdevice tapedump2
sp_addumpdevice "tape", tapedump2, "/dev/nrmt8", 625
```

Scheduling Backups of User Databases

A major task in developing a backup plan is determining how often to back up your databases. The frequency of your backups determines how much work you can lose in the event of a media failure. This section presents some guidelines about when to dump user databases and transaction logs.

Scheduling Routine Backups

Dump each user database just after you create it, to provide a base point, and on a fixed schedule thereafter. Daily backups of the transaction log and weekly backups of the database are the minimum recommended. Many installations with large and active databases make database dumps every day and transaction log dumps every half hour or hour.

Interdependent databases—databases where there are cross-database transactions, triggers, or referential integrity—should be backed up at the same time, during a period when there is no cross-database data modification activity. If one of these databases fails and needs to be reloaded, they should all be reloaded from these simultaneous dumps.

◆ ***WARNING!***

Always dump both databases immediately after adding, changing, or removing a cross-database constraint or dropping a table that contains a cross-database constraint.

Other Times to Back Up a Database

In addition to routine dumps, you should dump a database each time you upgrade a user database, create a new index, perform an unlogged operation, or run the **dump transaction with no_log** or **dump transaction with truncate_only** command.

Dumping a User Database After Upgrading

After upgrading a user database to the current release of Adaptive Server, you must dump the newly upgraded database to create a dump that is compatible with the current release. A **dump database** must occur on upgraded user databases before a **dump transaction** is permitted.

Dumping a Database After Creating an Index

When you add an index to a table, the **create index** command is recorded in the transaction log. As it fills the index pages with information, however, Adaptive Server does not log the changes.

If your database device fails after you create an index, the **load transaction** command may take as long to reconstruct the index as the **create index** command took to build it. To avoid lengthy delays, dump each database immediately after creating an index on one of its tables.

Dumping a Database After Unlogged Operations

Adaptive Server writes the data for the following commands directly to disk, adding no entries (or, in the case of **bcp**, minimal entries) in the transaction log:

- Non-logged **writetext**
- **select into** on a permanent table
- Fast bulk copy (**bcp**) into a table with no triggers or indexes

All changes are not recorded in the transaction log, so you cannot recover them from a transaction log dump or recover any changes made to the database after issuing one of these commands. To ensure that these commands are recoverable, issue a **dump database** command immediately after executing any of these commands.

Dumping a Database When the Log Has Been Truncated

The **dump transaction with truncate_only** and **dump transaction with no_log** commands remove transactions from the log without making a backup copy. To ensure recoverability, you must dump the database each time it is necessary for you to run either command because of lack of disk space. Adaptive Server prohibits you from copying the transaction log until you have done so. See "Using the Special dump transaction Options" on page 20-22 for more information about **dump transaction with truncate_only** and **dump transaction with no_log** restrictions.

If the **trunc log on chkpt** database option is set to **true**, and the transaction log contains 50 rows or more, Adaptive Server truncates the log when an automatic checkpoint occurs. Then, you must dump the entire database—not the transaction log—to ensure recoverability.

Scheduling Backups of *master*

Backups of the *master* database are used as part of the recovery procedure in case of a failure that affects the *master* database. If you do not have a current backup of *master*, you may have to reconstruct vital system tables at a time when you are under pressure to get your

databases up and running again. Be prepared—back up the *master* database **regularly and frequently**.

Dumping *master* After Each Change

Back up the *master* database with **dump database** each time you make a change to the database. Although you can restrict the creation of database objects in *master*, system procedures such as **sp_addlogin** and **sp_droplogin**, **sp_password**, and **sp_modifylogin** allow users to modify system tables in the database. Be sure to back up the *master* database frequently to record these changes.

Back up the *master* database after each command that affects disks, storage, databases, or segments. Always back up *master* after issuing any of the following commands or system procedures:

- **disk init**, **sp_addumpdevice**, or **sp_dropdevice**
- Disk mirroring commands
- The segment system procedure **sp_addsegment, sp_dropsegment**, or **sp_extendsegment**
- **create procedure** or **drop procedure**
- **sp_logdevice**
- **sp_configure**
- **create database** or **alter database**

Saving Scripts and System Tables

For further protection, save the scripts containing all of your **disk init**, **create database**, and **alter database** commands and make a hard copy of your *sysdatabases*, *sysusages*, and *sysdevices* tables each time you issue one of these commands.

Changes that result from these commands cannot be recovered automatically with **buildmaster**. If you keep your scripts—files containing Transact-SQL statements—you can run them to re-create the changes. Otherwise, you must reissue each command against the rebuilt *master* database.

You should also keep a hard copy of *syslogins*. When you recover *master* from a dump, compare the hard copy to your current version of the table to be sure that users retain the same user IDs.

For information on the exact queries to run against the system tables, see "Backing Up master and Keeping Copies of System Tables" on page 2-4.

Truncating the *master* Database Transaction Log

Since the *master* database transaction log is on the same database devices as the data, you cannot back up its transaction log separately. You cannot move the log of the *master* database. You must always use **dump database** to back up the master database. Use **dump transaction** with the **truncate_only** option periodically (for instance, after each database dump) to purge the transaction log of the *master* database.

Avoiding Volume Changes and Recovery

When you dump the *master* database, be sure that the entire dump fits on a single volume, unless you have more than one Adaptive Server that can communicate with your Backup Server. You must start Adaptive Server in single-user mode before loading the *master* database. This does not allow a separate user connection to respond to Backup Server's volume change messages during the load. Since *master* is usually small in size, placing its backup on a single tape volume is typically not a problem.

Scheduling Backups of the *model* Database

Keep a current database dump of the *model* database. Each time you make a change to the *model* database, make a new backup. If *model* is damaged and you do not have a backup, you must redo all the changes you have made in order to restore *model*.

Truncating the *model* Database's Transaction Log

model, like *master*, stores its transaction log on the same database devices as the data. You must always use **dump database** to back up the *model* database and **dump transaction** with the **truncate_only** option to purge the transaction log after each database dump.

Scheduling Backups of the *sybsystemprocs* Database

The *sybsystemprocs* database stores only system procedures. It is very easy to restore this database by running the **installmaster** script, unless you make changes to the database.

If you change permissions on some system procedures, or create your own system procedures in *sybsystemprocs*, your two recovery choices are:

- Run the **installmaster** script, and then redo all of your changes by re-creating your procedures or by reexecuting the **grant** and **revoke** commands.
- Back up *sybsystemprocs* each time you make a change to it.

Both of these recovery options are described in Chapter 22, "Restoring the System Databases."

Like other system databases, *sybsystemprocs* stores its transaction log on the same device as the data. You must always use **dump database** to back up the *sybsystemprocs* database. By default, the **trunc log on chkpt** option is set to **true** (on) in *sybsystemprocs*, so you should not need to truncate the transaction log. If you change this database option, be sure to truncate the log when you dump the database.

If you are running on a UNIX system or on a PC, and you have only one Adaptive Server that can communicate with your Backup Server, be sure that the entire dump of *sybsystemprocs* fits on a single dump device. Signaling volume changes requires the **sp_volchanged** system procedure, and you cannot use this procedure on a server while the *sybsystemprocs* database is in the process of recovery.

Configuring Adaptive Server for Simultaneous Loads

Adaptive Server can perform multiple load and dump commands simultaneously. The process of loading a database requires one 16K buffer for each active database load. By default, Adaptive Server is configured for six simultaneous loads. If you need to perform more loads simultaneously, a System Administrator can increase the value of **number of large i/o buffers**:

```
sp_configure "number of large i/o buffers", 12
```

This configuration parameter is static and requires a restart of Adaptive Server. See "number of large i/o buffers" on page 11-21 for more information. These buffers are not used for **dump** commands or for **load transaction**.

Gathering Backup Statistics

Once you have finished reading this chapter, read Chapter 21, "Backing Up and Restoring User Databases," and Chapter 22, "Restoring the System Databases." Then, practice using the backup and load commands.

Use **dump database** to make several practice backups of an actual user database and **dump transaction** to back up a transaction log. Recover the database with the **load database** command and apply successive transaction log dumps with the **load transaction** command.

Keep statistics on how long each dump and load takes and how much space it requires. The more closely you approximate real-life backup conditions, the more meaningful your predictions will be.

Once you have developed and tested your backup procedures, commit them to paper. Determine a reasonable backup schedule and adhere to it. If you develop, document, and test your backup procedures ahead of time, you will be much better prepared to get your databases online when disaster strikes.

21 Backing Up and Restoring User Databases

Regular and frequent backups are your only protection against database damage that results from failure of your database devices.

This chapter is the second of a four-chapter unit on backup and recovery. It describes how to use the **dump** and **load** commands for backup, recovery, and log truncation. It includes the following topics:

- Dump and Load Command Syntax 21-2
- Specifying the Database and Dump Device 21-5
- Specifying a Remote Backup Server 21-10
- Specifying Tape Density, Block Size, and Capacity 21-11
- Specifying the Volume Name 21-14
- Identifying a Dump 21-15
- Specifying Additional Dump Devices: the stripe on Clause 21-17
- Tape Handling Options 21-20
- Overriding the Default Message Destination 21-24
- Getting Information About Dump Files 21-25
- Copying the Log After a Device Failure 21-28
- Truncating a Log That Is Not on a Separate Segment 21-30
- Truncating the Log in Early Development Environments 21-30
- Truncating a Log That Has No Free Space 21-31
- Responding to Volume Change Requests 21-34
- Recovering a Database: Step-by-Step Instructions 21-39
- Upgrading User Database Dumps 21-46
- Cache Bindings and Loading Databases 21-49
- Cross-Database Constraints and Loading Databases 21-51

Table 21-1 lists additional sources of information about backup and recovery:

Table 21-1: Further information about backup and recovery

For More Information About	See
Backup and recovery issues to address before production	Chapter 20, "Developing a Backup and Recovery Plan"
Backing up and restoring the system databases	Chapter 22, "Restoring the System Databases"
Using thresholds to automate backups	Chapter 23, "Managing Free Space with Thresholds"
Backing up objects on remote servers when Component Integration Services is enabled	*Component Integration Services User's Guide*

Dump and Load Command Syntax

The **dump database**, **dump transaction**, **load database**, and **load transaction** commands have parallel syntax. Routine dumps and loads require the name of a database and at least one dump device. The commands can also include the following options:

- **at** *server_name* to specify the remote Backup Server
- **density**, **blocksize**, and **capacity** to specify tape storage characteristics
- **dumpvolume** to specify the volume name of the ANSI tape label
- **file =** *file_name* to specify the name of the file to dump to or load from
- **stripe on** *stripe_device* to specify additional dump devices
- **dismount**, **unload**, **init**, and **retaindays** to specify tape handling
- **notify** to specify whether Backup Server messages are sent to the **client** that initiated the dump or load or to the **operator_console**

Table 21-2 shows the syntax for routine database and log dumps and for dumping the log after a device failure. It indicates what type of information is provided by each part of the **dump database** or **dump transaction** statement.

Table 21-2: Syntax for routine dumps and log dumps after device failure

Information Provided	Task	
	Routine Database or Log Dump	**Log Dump After Device Failure**
Command	**dump** {**database** | **transaction**}	**dump transaction**
Database name	*database_name*	*database_name*
Dump device	**to** *stripe_device*	**to** *stripe_device*
Remote Backup Server	[**at** *server_name*]	[**at** *server_name*]
Tape device characteristics	[**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*]	[**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes]*
Volume name	[**, dumpvolume =** *volume_name*]	[**, dumpvolume =** *volume_name]*
File name	[**, file** = *file_name*]	[**, file =** *file_name*]
Characteristics of additional devices (up to 31 devices; one set per device)	[**stripe on** *stripe_device*	[**stripe on** *stripe_device*
	[**at** *server_name]*	[**at** *server_name]*
	[**density =** *density*,	[**density =** *density*,
	blocksize = *number_bytes*,	**blocksize =** *number_bytes*,
	capacity = *number_kilobytes*, **file =** *file_name*, **dumpvolume =** *volume_name*]]...	**capacity =** *number_kilobytes*, **file =** *file_name*, **dumpvolume** = *volume_name*]]...
Options that apply to entire dump	[**with** { **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **file =** *file_name*, [**nodismount** | **dismount**],	[**with** { **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **file =** *file_name*, [**nodismount** | **dismount**],
	[**nounload** | **unload**],	[**nounload** | **unload**],
	[**retaindays =** *number_days*],	[**retaindays =** *number_days*],
	[**noinit** | **init**], **file =** *file_name*, **dumpvolume =** *volume_name*	[**noinit** | **init**], **file =** *file_name*, **dumpvolume =** *volume_name*,
Do not truncate log		**no_truncate**
Message destination	[**, notify =** {**client** | **operator_console**}] }]	[**, notify =** {**client** | **operator_console**}] }]

Table 21-3 shows the syntax for loading a database, applying transactions from the log, and returning information about dump headers and files:

Table 21-3: Syntax for load commands

Information Provided	Task	
	Load Database or Apply Recent Transactions	**Return Header or File Information but Do Not Load Backup**
Command	**load** {**database** | **transaction**}	**load** {**database** | **transaction**}
Database name	*database_name*	*database_name*
Dump device	**from** *stripe_device*	**from** *stripe_device*
Remote Backup Server	[**at** *server_name*]	[at *server_name*]
Tape device characteristics	[**density =** *density*, **blocksize =** *number_bytes*]	[**density =** *density*, **blocksize =** *number_bytes*]
Volume name	[, **dumpvolume =** *volume_name*]	[, **dumpvolume =** *volume_name*]
File name	[, **file =** *file_name*]	[, **file =** *file_name*]
Characteristics of additional devices (up to 31 devices; one set per device)	[**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **file =** *file_name*, **dumpvolume =** *volume_name*]]...	[**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **file =** *file_name*, **dumpvolume =** *volume_name*]]...
Tape handling	[**with**{ [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**]	[**with**{ [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**]
Provide header information		[, **headeronly**]
List dump files		[, **listonly** [= **full**]]
Message destination	[, **notify =** {**client** | **operator_console**}]}]	[, **notify =** {**client** | **operator_console**}]}]

Table 21-4 shows the syntax for truncating a log:

- That is not on a separate segment
- Without making a backup copy

- With insufficient free space to successfully complete a **dump transaction** or **dump transaction with truncate_only** command

Table 21-4: Special dump transaction options

Information Provided	Task		
	Truncate Log on Same Segment as Data	**Truncate Log Without Making a Copy**	**Truncate Log with Insufficient Free Space**
Command	**dump transaction**	**dump transaction**	**dump transaction**
Database name	*database_name*	*database_name*	*database_name*
Do Not Copy Log	**with truncate_only**	**with truncate_only**	**with no_log**

The remainder of this chapter provides greater detail about the information specified in dump and load commands and volume change messages. Routine dumps and loads are described first, followed by log dumps after device failure and the special syntax for truncating logs without making a backup copy.

For information about the permissions required to execute the dump and load commands, refer to "Designating Responsibility for Backups" on page 20-25.

Specifying the Database and Dump Device

At a minimum, all dump and load commands must include the name of the database being dumped or loaded. Commands that dump or load data (rather than just truncating a transaction log) must also include a dump device.

Table 21-5 shows the syntax for backing up a database or log, and loading a database or log.

.

Table 21-5: Indicating the database name and dump device

	Backing Up a Database or Log	Loading a Database or Log
Database name Dump device	**dump** {**database** | **tran**} *database_name* **to** *stripe_device*	**load** {**database** | **tran**} *database_name* **from** *stripe_device*
	[**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	[**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes* **dumpvolume =** *volume_name* **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

Rules for Specifying Database Names

The database name can be specified as a literal, a local variable, or a parameter to a stored procedure.

If you are loading a database from a dump:

- The database must exist. You can create a database with the **for load** option of **create database**, or load it over an existing database. Loading a database always overwrites all the information in the existing database.
- You do not need to use the same database name as the name of the database you dumped. For example, you can dump the *pubs2* database, create another database called *pubs2_archive*, and load the dump into the new database.

◆ ***WARNING!***

You should never change the name of a database that contains primary keys for references from other databases. If you must load a dump from such a database and provide a different name, first drop the references to it from other databases.

Rules for Specifying Dump Devices

Consider the following rules when specifying a dump device:

- You can specify the dump device as a literal, a local variable, or a parameter to a stored procedure.
- You cannot dump to or load from the "null device" (on UNIX, */dev/null*; on OpenVMS, any device name beginning with NL; not applicable to PC platforms).
- When dumping to or loading from a local device, you can use any of the following forms to specify the dump device:
  - An absolute path name
  - A relative path name
  - A logical device name from the *sysdevices* system table

  The Backup Server resolves relative path names using Adaptive Server's current working directory.
- When dumping or loading over the network:
  - You must specify the absolute path name of the dump device. You cannot use a relative path name or a logical device name from the *sysdevices* system table.
  - The path name must be valid on the machine on which the Backup Server is running.
  - If the name includes any characters except letters, numbers or the underscore (_), you must enclose it in quotes.

Examples

The following examples use a single tape device for dumps and loads. (It is not necessary to use the same device for dumps and loads.)

On UNIX:

```
dump database pubs2 to "/dev/nrmt4"
load database pubs2 from "/dev/nrmt4"
```

On OpenVMS:

```
dump database pubs2 to "MTA0:"
load database pubs2 from "MTA0:"
```

On Windows NT:

```
dump database pubs2 to "\\.\tape0"
load database pubs2 from "\\.\tape0"
```

You can also dump to an operating system file. The following example is for Windows NT:

```
dump database pubs2 to "d:\backups\backup1.dat"
load database pubs2 from "d:\backupbackup1.dat"
```

Tape Device Determination by Backup Server

To become less device-dependent and more flexible, Backup Server provides a method for determining the tape device characteristics (tape positioning for read, close, append, I/O size, file marks, and ability to overwrite a tape mark) for a dump operation.

When you issue a **dump database** or **dump transaction** command, Backup Server checks to see if the device type of the specified dump device is known (supplied and supported internally) by Adaptive Server. If the device is not a known type, Backup Server checks the tape configuration file (default location is *$SYBASE/backup_tape.cfg)* for the device configuration. If the configuration is found, the **dump** command proceeds.

If the configuration is not found in the tape device configuration file, the **dump** command fails with the following error message:

```
Device not found in configuration file. INIT needs
to be specified to configure the device.
```

This means the device needs to be configured. To configure the device, issue the **dump database** or **dump transaction** with **init** qualifier. Using operating system calls, Backup Server attempts to determine the device's characteristics; if successful, it stores the device characteristics in the tape configuration file.

If Backup Server cannot determine the dump device characteristics, it defaults to one dump per tape. The device cannot be used if the configuration fails to write at least one dump file.

Tape configuration by Backup Server applies only to UNIX platforms.

Tape Device Configuration File

Format

The tape device configuration file contains tape device information that is used only by the **dump** command.

The format of the file is one tape device entry per line. Fields are separated by blanks or tabs.

Creation

This file is not provided at installation or upgrade time. It is created only when Backup Server is ready to write to it (**dump database** or **dump transaction** with **init**). When Backup Server tries to write to this file for the first time, the following warning message is issued:

```
Warning, unable to open device configuration file
for reading. Operating system error. No such file
or directory.
```

Ignore this message. Backup Server gives this warning and then creates the file and writes the configuration information to it.

Manual Editing

Since Backup Server updates the configuration file, the only user interaction with the file occurs when the user receives the following error message:

```
Device does not match the current configuration.
Please reconfigure this tape device by removing
the configuration file entry and issuing a dump
with the INIT qualifier.
```

This means that the tape hardware configuration changed for a device name. Delete the line entry for that device name and issue a **dump** command, as instructed.

Default Location

The default path name for the configuration file is *$SYBASE/backup_tape.cfg*. You can change the default location with the Sybase installation utilities. See the installation documentation for your platform for more information.

Specifying a Remote Backup Server

Use the **at** *server_name* clause to send dump and load requests over the network to a Backup Server running on another machine.

Table 21-6 shows the syntax for dumping or loading from a remote Backup Server:

Table 21-6: Dumping to or loading from a remote Backup Server

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device*	**load** {**database** | **tran**} *database_name* **from** *stripe_device*
Remote Backup Server	[**at** *server_name*]	[**at** *server_name*]
	[**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	[**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

This is ideal for installations that use a single machine with multiple tape devices for all backups and loads. Operators can be stationed at these machines, ready to service all tape change requests.

The *server_name* must appear in the interfaces file on the computer where Adaptive Server is running, but does not need to appear in the *sysservers* table. The *server_name* must be the same in both the local and the remote interfaces file.

The following examples dump to and load from the remote Backup Server REMOTE_BKP_SERVER:

```
dump database pubs2
        to "/dev/nrmt0" at REMOTE_BKP_SERVER

load database pubs2
        from "/dev/nrmt0" at REMOTE_BKP_SERVER
```

Specifying Tape Density, Block Size, and Capacity

In most cases, the Backup Server uses a default tape density and block size that are optimal for your operating system; **we recommend that you use them**. The **density** and **blocksize** options allow you to override these defaults when they are not appropriate for particular devices. The **capacity** option allows you to specify tape capacity on platforms that do not reliably detect the end-of-tape marker.

You can specify a density, block size, and capacity for each dump device. You can also specify the **density**, **blocksize**, and **capacity** options in the **with** clause for all dump devices. Characteristics that are specified for an individual tape device take precedence over those specified in the **with** clause.

Table 21-7 shows the syntax for specifying the tape density, block size, and capacity:

Table 21-7: Specifying tape density, block size, and capacity

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to stripe_device** [**at** *server_name*]	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*]
Characteristics of a Single Tape Device	[**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*,	[**density =** *density*, **blocksize =** *number_bytes*,
	dumpvolume = *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device*] [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...]	**dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device*] [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...]
Characteristics of All Dump Devices	[**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*,	[**with**{ **density =** *density*, **blocksize =** *number_bytes*,
	dumpvolume = *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	**dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

The following sections provide greater detail about the **density**, **blocksize**, and **capacity** options.

Overriding the Default Density

The dump and load commands use the default tape density for your operating system. In most cases, this is the optimal density for tape dumps.

When dumping to tape on OpenVMS systems, you can override the default density with the **density** = *density* option. Valid densities are **800**, **1600**, **6250**, **6666**, **10000**, and **38000**. Not all densities are valid for all tape drives; specify a value that is correct for your drive.

This option has no effect on OpenVMS tape loads or on UNIX and PC platform dumps or loads.

➤ ***Note***

Specify tape density only when using the **init** tape handling option. For more information on this option, see the section "Reinitializing a Volume Before a Dump" on page 21-22.

Overriding the Default Block Size

By default, the dump and load commands choose the "best" block size for your operating system. Wherever possible, use these defaults.

You can use the **blocksize** = *number_bytes* option to override the default block size for a particular dump device. The block size must be at least one database page (2048 bytes) and must be an exact multiple of the database page size.

For OpenVMS systems, you can specify a block size only for dumps. Use a block size of less than or equal to 55,296.

For UNIX systems, you can specify a block size on both dump and load commands. When loading a dump, you must specify the same block size that was used to make the dump.

Specifying Tape Capacity for Dump Commands

By default, OpenVMS systems write until they reach the physical end-of-tape marker, and then signal that a volume change is required. For UNIX platforms that cannot reliably detect the end-of-tape marker, you must indicate how many kilobytes can be dumped to a tape.

If you specify the physical path name of the dump device, you must include the **capacity** = *number_kilobytes* parameter in the dump command. If you specify the logical dump device name, the Backup Server uses the *size* parameter stored in the *sysdevices* system table, unless you override it with the **capacity** = *number_kilobytes* parameter.

The specified capacity must be at least five database pages (each page requires 2048 bytes). We recommend that you specify a capacity that is slightly below the capacity rated for your device.

A general rule for calculating capacity is to use 70 percent of the manufacturer's maximum capacity for the device, and allow 30 percent for overhead (inter-record gaps, tape marks, and so on). This rule works in most cases, but may not work in all cases because of differences in overhead across vendors and across devices.

Specifying the Volume Name

Use the **with dumpvolume =** *volume_name* option to specify the volume name. The **dump database** and **dump transaction** commands write the volume name to the SQL tape label. The **load database** and **load transaction** commands check the label. If the wrong volume is loaded, Backup Server generates an error message.

You can specify a volume name for each dump device. You can also specify a volume name in the **with** clause for all devices. Volume names specified for individual devices take precedence over those specified in the **with** clause.

Table 21-8 shows the syntax for specifying a volume name:

Table 21-8: Specifying the volume name

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*,	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*,
Volume name for single device	**dumpvolume =** *volume_name*,	**dumpvolume =** *volume_name*,
	file = *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*,	**file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*]...] [**with** { **density =** *density*, **blocksize =** *number_bytes*,
Volume name for all devices	**dumpvolume =** *volume_name*,	**dumpvolume =** *volume_name*,

Table 21-8: Specifying the volume name (continued)

Backing Up a Database or Log	Loading a Database or Log
file = *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	**file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

Loading from a Multifile Volume

When you load a database dump from a volume that contains multiple dump files, specify the dump file name. If you omit the dump file name and specify only the database name, Backup Server loads the first dump file into the specified database. For example, entering the following command loads the first dump file from the tape into *pubs2*, regardless of whether that dump file contains data from *pubs2*:

```
load database pubs2 from "/dev/rdsk/clt3d0s6"
```

To avoid this problem, specify a unique dump file name each time you dump or load data. To get information about the dump files on a given tape, use the **listonly = full** option of the **load database** command.

Identifying a Dump

When you dump a database or transaction log, Backup Server creates a default file name for the dump by concatenating the:

- Last 7 characters of the database name
- 2-digit year number
- 3-digit day of the year (1–366)
- Number of seconds since midnight, in hexadecimal

You can override this default using the **file =** *file_name* option. The file name cannot exceed 17 characters and must conform to the file naming conventions for your operating system.

You can specify a file name for each dump device. You can also specify a file name for all devices in the **with** clause. File names specified for individual devices take precedence over those specified in the **with** clause.

Table 21-9 shows the syntax for specifying the name of a dump:

Table 21-9: Specifying the file name for a dump

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*,	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*,
File name for single device	**file =** *file_name*]	**file =** *file_name*]
	[**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*,	[**stripe on** *stripe_device*] [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*,
File name for all devices	**file =** *file_name*,	**file =** *file_name*,
	[**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	[**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

The following examples dump the transaction log for the *publications* database without specifying a file name. The default file name, *cations930590E100*, identifies the database and the date and time the dump was made:

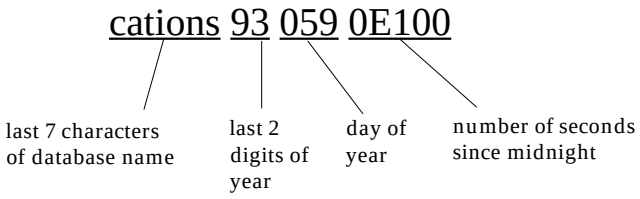


Figure 21-1: File-naming convention for database and transaction log dumps

Backup Server sends the file name to the default message destination or to the **notify** location for the dump command. Be sure to label each backup tape with the volume name and file name before storing it.

When you load a database or transaction log, you can use the **file =** *file_name* clause to specify which dump to load from a volume that contains multiple dumps:

When loading the dump from a multifile volume, you must specify the correct file name:

```
dump tran publications
    to "/dev/nrmt3"
load tran publications
    from "/dev/nrmt4"
    with file = "cations930590E100"
```

The following examples use a user-defined file-naming convention. The 15-character file name, *mydb97jul141800*, identifies the database (*mydb*), the date (July 14, 1997), and the time (18:00, or 6:00 p.m.) that the dump was made. Using the load command advances the tape to *mydb97jul141800* before loading:

```
dump database mydb
    to "/dev/nrmt3"
    with file = "mydb97jul141800"

load database mydb
    from "/dev/nrmt4"
    with file = "mydb97jul141800"
```

Specifying Additional Dump Devices: the *stripe on* Clause

Dump striping allows you to use multiple dump devices for a single dump or load command. Use a separate **stripe on** clause to specify the name (and, if desired, the characteristics) of each device.

Each dump or load command can have up to 31 **stripe on** clauses (for a maximum of 32 dump devices):

Table 21-10 shows the syntax for using more than one dump device:

Table 21-10: Using more than one dump device

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*]	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*]
Characteristics of an additional tape device (one set per device; up to 31 devices)	[**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...]	[**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...]
	[**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**], [**notify =** {**client** | **operator_console**}]}]	[**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], [**notify =** {**client** | **operator_console**}]}]

Dumping to Multiple Devices

The Backup Server divides the database into approximately equal portions and sends each portion to a different device. Dumps are made concurrently on all devices, reducing the time required to dump an individual database or transaction log. Because each tape stores only a portion of the database, it is less likely that new will have to be mounted on a particular device.

◆ ***WARNING!***

Do not dump the *master* database to multiple tape devices. When loading the *master* database from tape or other removable media, you cannot change volumes unless you have another Adaptive Server that can respond to volume change messages.

Loading from Multiple Devices

You can use up to 32 devices to load a database or transaction log. Using multiple devices decreases both the time required for the load and the likelihood of having to mount multiple tapes on a particular device.

Using Fewer Devices to Load Than to Dump

You can load a database or log even if one of your dump devices becomes unavailable between the dump and load. Specify fewer stripe clauses in the load command than you did in the dump command.

➤ ***Note***

When you dump and load over the network, you must use the same number of drives for both operations.

The following examples use three devices to dump a database but only two to load it:

```
dump database pubs2 to "dev/nrmt0"
    stripe on "/dev/nrmt1"
    stripe on "/dev/nrmt2"

load database pubs2 from "/dev/nrmt0"
    stripe on "/dev/nrmt1"
```

After the first two tapes are loaded, a message notifies the Operator to load the third.

You can also dump a database to multiple operating system files. The following example is for Windows NT:

```
dump database pubs2 to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
```

```
load database pubs2 from "/dev/nrmt0"
stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
```

Specifying the Characteristics of Individual Devices

Use a separate **at** *server_name* clause for each stripe device attached to a remote Backup Server. If you do not specify a remote Backup Server name, the local Backup Server looks for the dump device on the local machine. If necessary, you can also specify separate tape device characteristics (**density, blocksize, capacity, dumpvolume**, and **file**) for individual stripe devices.

The following examples use three dump devices, each attached to the remote Backup Server REMOTE_BKP_SERVER:

On UNIX:

```
dump database pubs2
        to "/dev/nrmt0" at REMOTE_BKP_SERVER
        stripe on "/dev/nrmt1" at REMOTE_BKP_SERVER
        stripe on "/dev/nrmt2" at REMOTE_BKP_SERVER
```

On OpenVMS:

```
dump database pubs2
    to "MTA0:" at REMOTE_BKP_SERVER
    stripe on "MTA1:" at REMOTE_BKP_SERVER
    stripe on "MTA2:" at REMOTE_BKP_SERVER
```

Tape Handling Options

The tape handling options, which appear in the **with** clause, apply to all devices used for the dump or load. They include:

- **nodismount** to keep the tape available for additional dumps or loads
- **unload** to rewind and unload the tape following the dump or load
- **retaindays** to protect files from being overwritten
- **init** to reinitialize the tape rather than appending the dump files after the last end-of-tape mark

Table 21-11 shows the syntax for tape handling options:

Table 21-11: Tape handling options

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name]* [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*,	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes* **dumpvolume =** *volume_name* **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*,
Tape Handling Options	[**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**],	**nodismount** | **dismount**], [**nounload** | **unload**],
	[**notify =** {**client** | **operator_console**}]}]	[**notify =** {**client** | **operator_console**}]}]

Specifying Whether to Dismount the Tape

On platforms that support logical dismounts, such as OpenVMS, tapes are dismounted when a dump or load completes. Use the **nodismount** option to keep the tape mounted and available for additional dumps or loads. This command has no effect on UNIX or PC systems.

Rewinding the Tape

By default, both dump and load commands use the **nounload** tape handling option.

On UNIX systems, this prevents the tape from rewinding after the dump or load completes. This allows you to dump additional databases or logs to the same volume or to load additional databases or logs from that volume. Use the **unload** option for the last dump on the tape to rewind and unload the tape when the command completes.

On OpenVMS systems, tapes are always rewound after a dump or load completes. Use the **unload** option to unthread the tape and eject it from the drive. (This action is equivalent to the **/UNLOAD** qualifier for the OpenVMS **DISMOUNT** command.)

Protecting Dump Files from Being Overwritten

The **tape retention in days** configuration parameter specifies the number of days that must elapse between the creation of a tape file and the time at which you can overwrite it with another dump. This server-wide variable, which is set with the **sp_configure** system procedure, applies to all dumps requested from a single Adaptive Server.

Use the **retaindays** = ***number_days*** option to override the **tape retention in days** parameter for a single database or transaction log dump. The number of days must be a positive integer, or zero if the tape can be overwritten immediately.

➤ ***Note***

tape retention in days and **retaindays** are meaningful only for disk, 1/4-inch cartridge, and single-file media. On multifile media, the Backup Server checks only the expiration date of the first file.

Reinitializing a Volume Before a Dump

By default, each dump is appended to the tape following the last end-of-tape mark. Tape volumes are not reinitialized. This allows you to dump multiple databases to a single volume. (New dumps can be appended only to the last volume of a multivolume dump.)

Use the **init** option to overwrite any existing contents of the tape. If you specify **init**, the Backup Server reinitializes the tape **without** checking for:

- ANSI access restrictions
- Files that have not yet expired

- Non-Sybase data (“foreign” tapes on OpenVMS)

The default, **noinit**, checks for all three conditions and sends a volume change prompt if any are present.

The following example initializes two devices, overwriting the existing contents with the new transaction log dumps:

```
dump transaction pubs2
    to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    with init
```

You can also use the **init** option to overwrite an existing file, if you are dumping a database to an operating system file. The following example is for Windows NT:

```
dump transaction pubs2
    to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    with init
```

Dumping Multiple Databases to a Single Volume

Follow these steps to dump multiple databases to the same tape volume:

1. Use the **init** option for the first database. This overwrites any existing dumps and places the first dump at the beginning of the tape.
2. Use the default (**noinit** and **nounload**) option for subsequent databases. This places them one after the other on the tape.
3. Use the **unload** option for the last database on the tape. This rewinds and unloads the tape after the last database is dumped.

Figure 21-2 illustrates which options to use to dump three databases to a single tape volume.

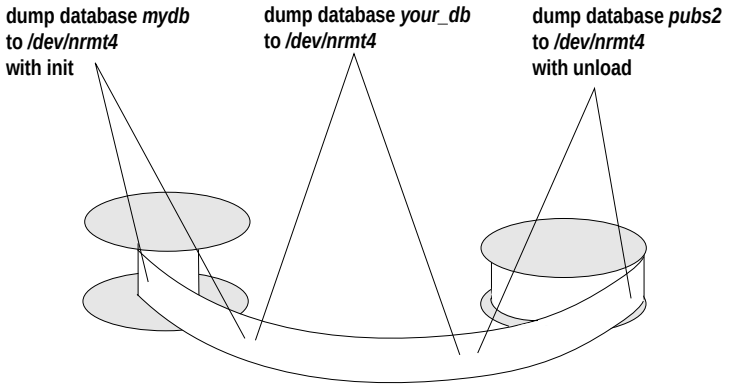


Figure 21-2: Dumping several databases to the same volume

Overriding the Default Message Destination

Backup Server messages inform the operator when to change tape volumes and how the dump or load is progressing. The default destination for these messages depends on whether the operating system offers an operator terminal feature.

The **notify** option, which appears in the **with** clause, allows you to override the default message destination for a dump or load.

For this option to work, the controlling terminal or login session from which Backup Server was started must remain active for as long as Backup Server is working; otherwise, the **sp_volchanged** message is lost.

On operating systems, such as OpenVMS, that offer an operator terminal feature, volume change messages are always sent to an operator terminal on the machine where the Backup Server is running. (OpenVMS routes messages to terminals that are enabled for TAPES, DISKS, or CENTRAL.) Use **notify = client** to route other Backup Server messages to the terminal session where the **dump** or **load** request initiated.

On systems such as UNIX that do not offer an operator terminal feature, messages are sent to the client that initiated the dump or load request. Use **notify = operator_console** to route messages to the terminal where the remote Backup Server was started.

Table 21-12 shows the syntax for overriding the default message destination:

Table 21-12: Overriding the default message destination

	Backing Up a Database or Log	Loading a Database or Log
	dump {**database** | **tran**} *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**],	**load** {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes* **dumpvolume =** *volume_name* **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**],
Message destination	[**notify =** {**client** | **operator_console**}]}]	[**notify =** {**client** | **operator_console**}]}]

Getting Information About Dump Files

If you are unsure of the contents of a tape, use the **with headeronly** or **with listonly** option of the load commands to request that information.

Table 21-13 shows the syntax for finding the contents of a tape:

Table 21-13: Listing dump headers or file names

	Listing Information About a Dump
	load {**database** | **tran**} *database_name* **from** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes* **dumpvolume =** *volume_name* **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**],
List header information List files on tape	[**headeronly** [, **file =** *filename*]], [**listonly** [**= full**]],
	[**notify =** {**client** | **operator_console**}]}]

➤ ***Note***

Neither **with headeronly** nor **with listonly** loads the dump files after displaying the report.

Requesting Dump Header Information

with headeronly returns the header information for a single file. If you do not specify a file name, **with headeronly** returns information about the first file on the tape.

The header indicates whether the dump is for a database or transaction log, the database ID, the file name, and the date the dump was made. For database dumps, it also shows the character set, sort order, page count, and next object ID. For transaction log dumps, it

shows the checkpoint location in the log, the location of the oldest **begin transaction** record, and the old and new sequence dates.

The following example returns header information for the first file on the tape and then for the file *mydb9229510945*:

```
load database mydb
    from "/dev/nrmt4"
    with headeronly

load database mydb
    from "/dev/nrmt4"
    with headeronly, file = "mydb9229510945"
```

Here is sample output from **headeronly**:

```
Backup Server session id is: 44. Use this value when executing
the 'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.

Backup Server: 4.28.1.1: Dumpfile name 'mydb9232610BC8 ' section
number 0001 mounted on device 'backup/SQL_SERVER/mydb.db.dump'

This is a database dump of database ID 5 from Nov 21 1992 7:02PM.

Database contains 1536 pages; checkpoint RID=(Rid pageid =
0x404; row num = 0xa); next object ID=3031; sort order ID=50,
status=0; charset ID=1.
```

Determining the Database, Device, File Name, and Date

with listonly returns a brief description of each dump file on a volume. It includes the name of the database, the device used to make the dump, the file name, the date and time the dump was made, and the date and time it can be overwritten. **with listonly = full** provides greater detail. Both reports are sorted by SQL tape label.

Following is sample output of a **load database** command **with listonly**:

```
Backup Server: 4.36.1.1: Device '/dev/nrst0':
File name: 'model9320715138  '
Create date & time: Monday, Jul 26, 1993, 23:58:48
Expiration date & time:  Monday, Jul 26, 1993, 00:00:00
Database name: 'model                          '
```

and sample output from **with listonly = full**:

```
Backup Server: 4.37.1.1: Device '/dev/nrst0':
Label id: 'HDR1'
File name:'model9320715138  '
Stripe count:0001
Device typecount:01
```

```
Archive volume number:0001
Stripe position:0000
Generation number:0001
Generation version:00
Create date & time:Monday, Jul 26, 1993, 23:58:48
Expiration date & time:Monday, Jul 26, 1993, 00:00:00
Access code:‘ ‘
File block count:000000
Sybase id string:
‘Sybase  ‘Reserved:‘       ‘
Backup Server: 4.38.1.1: Device ‘/dev/nrst0’:
Label id:‘HDR2’
Record format:‘F’
Max. bytes/block:55296
Record length:02048
Backup format version:01
Reserved:‘   ‘
Database name:‘model                          ‘
Buffer offset length:00
Reserved:‘                              ‘
```

After listing all files on a volume, the Backup Server sends a volume change request:

```
Backup Server: 6.30.1.2: Device /dev/nrst0: Volume cataloguing
complete.
Backup Server: 6.51.1.1: OPERATOR: Mount the next volume to
search.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
          @session_id = 5,
          @devname = ‘/dev/nrst0’,
          @action = { ‘PROCEED’ | ‘RETRY’ | ‘ABORT’ },
          @fname = ‘
```

The operator can mount another volume and signal the volume change with **sp_volchanged** or use **sp_volchanged** to terminate the search operation for all stripe devices.

Copying the Log After a Device Failure

Normally, the **dump transaction** command truncates the inactive portion of the log after copying it. Use the **with no_truncate** option to copy the log without truncating it.

The **no_truncate option** allows you to copy the transaction log after failure of the device that holds your data. It uses pointers in the *sysdatabases* and *sysindexes* system tables to determine the physical

location of the transaction log. It can be used only if your transaction log is on a separate segment and your *master* database is accessible.

◆ ***WARNING!***

Use no_truncate only if media failure makes your data segment inaccessible. Never use no_truncate on a database that is in use.

Copying the log **with no_truncate** is the first step described in "Recovering a Database: Step-by-Step Instructions" on page 21-39. To completely recover a database, follow the steps in that section.

Table 21-14 shows the syntax for copying a log after a device failure:

Table 21-14: Copying the log file after a device failure

	Copying with the *no_truncate* Option
	dump transaction *database_name* **to** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*] [**stripe on** *stripe_device* [**at** *server_name*] [**density =** *density*, **blocksize =** *number_bytes*, **capacity** = *number_kilobytes*, **dump volume =** *volume_name*, **file =** *file_name*] ...] [**with**{ **density =** *density*, **blocksize =** *number_bytes*, **capacity =** *number_kilobytes*, **dumpvolume =** *volume_name*, **file =** *file_name*, [**nodismount** | **dismount**], [**nounload** | **unload**], **retaindays =** *number_days*, [**noinit** | **init**],
Do not truncate log	**no_truncate**,
	[**notify =** {**client** | **operator_console**}]}]

You can use **no_truncate** with striped dumps, tape initialization, and remote Backup Servers. Here is an example:

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

Truncating a Log That Is Not on a Separate Segment

If a database does not have a log segment on a separate device from data segments, you cannot use **dump transaction** to copy the log and then truncate it. For these databases, you must:

1. Use the special **with truncate_only** option of **dump transaction** to truncate the log so that it does not run out of space
2. Use **dump database** to copy the entire database, including the log

Because it copies no data, the **with truncate_only** option requires only the name of the database:

```
dump transaction database_name with truncate_only
```

The following example dumps the database *mydb*, which does not have a log segment on a separate device from data segments, and then truncates the log:

```
dump database mydb to mydevice
dump transaction mydb with truncate_only
```

Truncating the Log in Early Development Environments

In early development environments, the transaction log is quickly filled by the process of creating, dropping, and re-creating stored procedures and triggers and checking integrity constraints. Recovery of data may be less important than ensuring that there is adequate space on database devices.

The **with truncate_only** option of **dump transaction** is often useful in these environments. It allows you to truncate the transaction log without making a backup copy:

```
dump transaction database_name with truncate_only
```

After you run **dump transaction with truncate_only,** Adaptive Server requires that you dump the database before you can run a routine log dump.

Truncating a Log That Has No Free Space

When the transaction log is very full, you may not be able to use your usual method to dump it. If you used **dump transaction** or **dump transaction with truncate_only**, and the command failed because of insufficient log space, use the special with **no_log option** of **dump transaction**:

```
dump transaction database_name with no_log
```

This option truncates the log without logging the dump transaction event. Because it copies no data, it requires only the name of the database.

◆ ***WARNING!***

Use dump transaction with no_log as a last resort, and use it only once after dump transaction with truncate_only fails. If you continue to load data after entering dump transaction with no_log, it is possible to fill the log completely, causing any further dump transaction commands to fail. Use the alter database command to allocate additional space to the database.

All occurrences of **dump tran with no_log** are reported in the Adaptive Server error log. The message includes the user ID of the user executing the command. Messages indicating success or failure are also sent to the error log. **no_log** is the only dump option that generates error log messages.

Dangers of Using *with truncate_only* and *with no_log*

with truncate_only and **with no_log** allow you to truncate a log that has become disastrously short of free space. Neither option provides a means to recover your databases.

After running **dump transaction with truncate_only** or **with no_log**, you have no way to recover transactions that have committed since the last routine dump.

◆ ***WARNING!***

Run dump database at the earliest opportunity to ensure that your data can be recovered.

The following example truncates the transaction log for *mydb* and then dumps the database:

```
dump transaction mydb
    with no_log
dump database mydb to ...
```

Providing Enough Log Space

Every use of **dump transaction...with no_log** is considered an error and is recorded in the server's error log. If you have created your databases with log segments on a separate device from data segments, written a last-chance threshold procedure that dumps your transaction log often enough, and allocated enough space to your log and database, you should not have to use this option.

However, some situations can still cause the transaction log to become too full, even with frequent log dumps. The **dump transaction** command truncates the log by removing all pages from the beginning of the log, up to the page preceding the page that contains an uncommitted transaction record (known as the oldest active transaction). The longer this active transaction remains uncommitted, the less space is available in the transaction log, since **dump transaction** cannot truncate additional pages.

This can happen in situations when applications with very long transactions modify tables in a database with a small transaction log, which indicates you should increase the size of the log. It also occurs when transactions inadvertently remain uncommitted for long periods of time, such as when an implicit **begin transaction** uses the chained transaction mode or when a user forgets to complete the transaction. You can determine the oldest active transaction in each database by querying the *syslogshold* system table.

The *syslogshold* Table

The *syslogshold* table is in the *master* database. Each row in the table represents either:

- The oldest active transaction in a database, or
- The Replication Server truncation point for the database's log.

A database may have no rows in *syslogshold*, a row representing one of the above, or two rows representing both of the above. For information about how a Replication Sever truncation point affects

the truncation of the database's transaction log, see the Replication Server documentation.

Querying *syslogshold* provides a "snapshot" of the current situation in each database. Since most transactions last for only a short time, the query's results may not be consistent. For example, the oldest active transaction described in the first row of *syslogshold* may finish before Adaptive Server completes the query of *syslogshold*. However, when several queries of *syslogshold* over time query the same row for a database, that transaction may prevent a **dump transaction** from truncating any log space.

Once the transaction log reaches the last-chance threshold, and **dump transaction** cannot free up space in the log, you can query *syslogshold* and *sysindexes* to identify the transaction holding up the truncation. For example:

```
select H.spid, H.name
from master..syslogshold H, threshdb..sysindexes I
where H.dbid = db_id("threshdb")
and I.id = 8
and H.page = I.first

spid    name
------  ------------------------------------
     8  $user_transaction

(1 row affected)
```

This query uses the object ID associated with *syslogs* (8) in the *threshdb* database to match the first page of its transaction log with the first page of the oldest active transaction in *syslogshold*.

You can also query *syslogshold* and *sysprocesses* in the *master* database to identify the specific host and application owning the oldest active transactions. For example:

```
select P.hostname, P.hostprocess, P.program_name,
   H.name, H.starttime
from sysprocesses P, syslogshold H
where P.spid = H.spid
and H.spid != 0
```

```
hostname hostprocess program_name name               starttime
-------- ----------- ------------ ------------------ ---------
eagle          15826 isql         $user_transaction  Sep  6 1997
                                                     4:29PM
hawk           15859 isql         $user_transaction  Sep  6 1997
                                                     5:00PM
condor         15866 isql         $user_transaction  Sep  6 1997
                                                     5:08PM

(3 rows affected)
```

Using the above information, you can notify or kill the user process owning the oldest active transaction and proceed with the **dump transaction**. You can also include the above types of queries in the threshold procedures for the database as an automatic alert mechanism. For example, you may decide that the transaction log should never reach its last-chance threshold. If it does, your last-chance threshold procedure (**sp_thresholdaction)** alerts you with information about the oldest active transaction preventing the transaction dump.

➤ ***Note***

The initial log records for a transaction may reside in a user log cache, which is not visible in *syslogshold* until the records are flushed to the log (for example, after a checkpoint).

For more information about the *syslogshold* system table, see the *Adaptive Server Reference Manual*. For information about the last-chance threshold and threshold procedures, see Chapter 23, "Managing Free Space with Thresholds."

Responding to Volume Change Requests

On UNIX and PC systems, use the **sp_volchanged** system procedure to notify the Backup Server when the correct volumes have been mounted. On OpenVMS systems, use the **REPLY** command.

To use **sp_volchanged**, log into any Adaptive Server that can communicate with both the Backup Server that issued the volume change request and the Adaptive Server that initiated the dump or load.

sp_volchanged Syntax

Use the following syntax for **sp_volchanged**:

```
sp_volchanged session_id, devname , action
   [ ,fname [, vname] ]
```

- Use the *session_id* and *devname* parameters specified in the volume change request.
- *action* specifies whether to **abort**, **proceed** with, or **retry** the dump or load.
- *fname* specifies which file to load. If you do not specify a file name with **sp_volchanged**, the Backup Server loads the **file** = *file_name* parameter of the load command. If neither **sp_volchanged** nor the load command specifies which file to load, the Backup Server loads the first file on the tape.
- The Backup Server writes the *vname* in the ANSI tape label when overwriting an existing dump, dumping to a brand new tape, or dumping to a tape whose contents are not recognizable. During loads, the Backup Server uses the *vname* to confirm that the correct tape has been mounted. If you do not specify a *vname* with **sp_volchanged**, the Backup Server uses the volume name specified in the dump or load command. If neither **sp_volchanged** nor the command specifies a volume name, the Backup Server does not check this field in the ANSI tape label.

Volume Change Prompts for Dumps

This section describes the volume change prompts that appear while you are dumping a database or transaction log. For each prompt, it indicates the possible Operator actions and the appropriate **sp_volchanged** response.

- `Mount the next volume to search.`

  When appending a dump to an existing volume, the Backup Server issues this message if it cannot find the end-of-file mark.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *devname*, **abort**
Mount a new volume and proceed with the dump	**sp_volchanged** *session_id*, *devname*, **proceed** [, *fname* [, *vname*]]

- `Mount the next volume to write.`

  The Backup Server issues this message when it reaches the end of the tape. This occurs when it detects the end-of-tape mark, dumps the number of kilobytes specified by the **capacity** parameter of the dump command, or dumps the *high* value specified for the device in the *sysdevices* system table.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *devname*, **abort**
Mount the next volume and proceed with the dump	**sp_volchanged** *session_id*, *devname*, **proceed** [, *fname* [, *vname*]]

- `Volume on device devname has restricted access (code access_code).`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a tape with ANSI access restrictions without specifying the **init** option.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *devname*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *devname*, **retry** [, *fname* [, *vname*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *devname*, **proceed** [, *fname* [, *vname*]]

- `Volume on device devname is expired and will be overwritten.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. During dumps to single-file media, Backup Server issues this message if you have not specified the **init**

option and the tape contains a dump whose expiration date has passed.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *devname*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *session_id*, **retry** [, *session_id* [, *session_id*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

- `Volume to be overwritten on 'devname' has not expired: creation date on this volume is creation_date, expiration date is expiration_date.`

  On single-file media, the Backup Server checks the expiration date of any existing dump unless you specify the **init** option. The Backup Server issues this message if the dump has not yet expired.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *session_id*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *session_id*, **retry** [, *session_id* [, *session_id*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

- `Volume to be overwritten on 'devname' has unrecognized label data.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try

to dump to a new tape or a tape with non-Sybase data without specifying the **init** option.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *session_id*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *session_id*, **retry** [, *session_id* [, *session_id*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

Volume Change Prompts for Loads

Following are the volume change prompts and possible operator actions during loads:

- `Dumpfile 'fname' section vname found instead of 'fname' section vname.`

  The Backup Server issues this message if it cannot find the specified file on a single-file medium.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *session_id*, **abort**
Mount another volume and try to load it	**sp_volchanged** *session_id*, *session_id*, **retry** [, *session_id* [, *session_id*]]
Load the file on the currently mounted volume, even though it is not the specified file (not recommended)	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

- `Mount the next volume to read.`

The Backup Server issues this message when it is ready to read the next section of the dump file from a multivolume dump.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *session_id*, **abort**
Mount the next volume and proceed with the load	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

- `Mount the next volume to search.`

  The Backup Server issues this message if it cannot find the specified file on multifile medium.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *session_id*, **abort**
Mount another volume and proceed with the load	**sp_volchanged** *session_id*, *session_id*, **proceed** [, *session_id* [, *session_id*]]

Recovering a Database: Step-by-Step Instructions

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption occurs, unless you are running **dbcc** commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. Adaptive Server marks the database as suspect and displays a warning message. If the disk storing the *master* database fails, users will not be able to log into the server, and users already logged in will not be able to perform any actions that access the system tables in *master*.

This section describes what to do when a database device fails. The recommended procedure consists of the following steps:

1. Get a current log dump of **every database on the device**.
2. Examine the space usage of **every database on the device**.
3. Once you have gathered this information for all databases on the device, drop each database.
4. Drop the failed device.
5. Initialize new devices.

6. Re-create the databases, one at a time.
7. Load the most recent database dump into each database.
8. Apply each transaction log dump in the order in which it was created.

These steps are described in detail in the following sections.

Getting a Current Dump of the Transaction Log

Use **dump transaction with no_truncate** to get a current transaction log dump **for each database on the failed device**. For example, to get a current transaction log dump of *mydb*:

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

Examining the Space Usage

The following steps are recommended to determine which devices your database uses, how much space is allocated on each device, and whether the space is used for data, log, or both. You can use this information when re-creating your databases to ensure that the log, data, and indexes reside on separate devices, and to preserve the scope of any user segments you have created.

➤ ***Note***

You can also use these steps to preserve segment mappings when moving a database dump from one server to another (on the same hardware and software platform).

If you do not use this information to re-create the device allocations for damaged databases, Adaptive Server will **remap** the *sysusages* table after **load database** to account for discrepancies. This means that the database's system-defined and user-defined segments no longer match the appropriate device allocations. Incorrect information in *sysusages* can result in the log being stored on the same devices as the data, even if the data and the log were separate before recovery. It can also change user-defined segments in unpredictable ways, and can result in a database that cannot be created using a standard **create database** command.

Follow these steps to examine and record the device allocations for all damaged databases:

1. In *master*, use the following query to examine the device allocations and uses for the damaged database:

   ```
   select segmap, size from sysusages
       where dbid = db_id("database_name")
   ```

2. Examine the output of the query. Each row with a *segmap* of "3" represents a data allocation; each row with a *segmap* of "4" represents a log allocation. Higher values indicate user-defined segments; treat these as data allocations, to preserve the scope of these segments. The *size* column indicates the number of 2K blocks of data. To find the number of megabytes, divide by 512. Note the order, use, and size of each disk piece.

   For example, this output:

   ```
   segmap      size
   -------     --------
        3        10240
        3         5120
        4         5120
        8         1024
        4         2048
   ```

   translates into the sizes and uses described in Table 21-15.

Table 21-15: Sample device allocation

Device Allocation	Megabytes
Data	20
Data	10
Log	10
Data (user-defined segment)	2
Log	4

➤ ***Note***

If the *segmap* column contains 7's, your data and log are on the same device, and you can recover only up to the point of the most recent database dump. **Do not** use the **log on** option to **create database**. Just be sure that you allocate as much (or more) space than the total reported from *sysusages*.

3. Run **sp_helpdb** ***database_name*** for the database. This query lists the devices on which the data and logs are located:

```
name       db_size owner   dbid    created
-------    ------- ------  ----    -----------
mydb       46.0 MB sa        15    Apr  9 1991

status          device_fragments    size          usage
--------------  ----------------    -----         ------------
no options set  datadev1            20 MB         data only
                datadev2            10 MB         data only
                datadev3             2 MB         data only
                logdev1             10 MB         log only
                logdev1              4 MB         log only
```

Dropping the Databases

Once you have performed the preceding steps **for all databases on the failed device**, use the **drop database** command to drop each database.

➤ ***Note***

If tables in other databases contain references to any tables in the database you are trying to drop, you must remove the referential integrity constraints with **alter table** before you can drop the database.

If the system reports errors because the database is damaged when you issue the **drop database** command, use the **dropdb** option of the **dbcc dbrepair** command:

```
dbcc dbrepair (mydb, dropdb)
```

See the *Troubleshooting Guide* for more information about **dbcc dbrepair.**

Dropping the Failed Devices

After you have dropped each database, use **sp_dropdevice** to drop the failed device. See the *Adaptive Server Reference Manual* for information on this system procedure.

Initializing New Devices

Use **disk init** to initialize the new database devices. See Chapter 6, “Initializing Database Devices,” for complete information.

Re-Creating the Databases

Use the following steps to re-create each database using the segment information you collected earlier.

➤ *Note*

If you chose not to gather information about segment usage, use **create database...for load** to create a new database that is at least as large as the original.

1. Use the **create database** command with the **for load** option. Duplicate all device fragment mappings and sizes for each row of your database from the *sysusages* table, **up to and including the first log device**. Be sure to use the order of the rows as they appear in *sysusages*. (The results of **sp_helpdb** are in alphabetical order by device name, not in order of allocation.) For example, to re-create the *mydb* database allocations shown in Table 21-15 on page 21-41, enter the command:

```
create database mydb
    on datadev1 = 20,
       datadev2 = 10
log on logdev1 = 10
for load
```

➤ *Note*

create database...for load temporarily locks users out of the newly created database, and **load database** marks the database offline for general use. This prevents users from performing logged transactions during recovery.

2. Use the **alter database** command with the **for load** option to re-create the remaining entries, in order. Remember to treat device allocations for user segments as you would data allocations.

   In this example, to allocate more data space on *datadev3* and more log space on *logdev1*, the command is:

```
alter database mydb
     on datadev3 = 2
log on logdev1=4
for load
```

Loading the Database

Reload the database using **load database**. If the original database stored objects on user-defined segments (*sysusages* reports a *segmap* greater than 7) and your new device allocations match those of the dumped database, Adaptive Server preserves the user segment mappings.

If you did not create the new device allocations to match those of the dumped database, Adaptive Server will remap segments to the available device allocations. This remapping may also mix log and data on the same physical device.

➤ ***Note***

If an additional failure occurs while a database is being loaded, Adaptive Server does not recover the partially loaded database, and notifies the user. You must restart the database load by repeating the **load** command.

Loading the Transaction Logs

Use **load transaction** to apply transaction log backups **in the same sequence in which they were made**. Load the most current dump last.

Adaptive Server checks the timestamps on each dumped database and transaction log. If the dumps are loaded in the wrong order, or if user transactions have modified the transaction log between loads, the load fails.

Once you have brought a database up to date, use **dbcc** commands to check its consistency.

Loading a Transaction Log to a Point in Time

You can recover a database up to a specified point in time in its transaction log. To do so, use the **until_time** option of the **load transaction** command. This feature is useful if, for example, a user inadvertently drops an important table; you can use the **until_time** option to recover the changes made to the database containing the table up to a time just before the table was dropped.

To use the **until_time** option effectively after data has been destroyed, you must know the exact time the error occurred. You can find this by issuing a **select getdate** at the time of the error. For example, suppose

a user accidentally drops an important table, and then a few minutes later you get the current time in milliseconds:

```
select convert(char(26), getdate(), 109)

--------------------------
Mar 26 1997 12:45:59:650PM
```

After dumping the transaction log containing the error and loading the most recent database dump, load the transaction logs that were created after the database was last dumped. Then, load the transaction log containing the error by using **until_time**; for example:

```
load transaction employees_db
from "/dev/nrmt5"
with until_time = "Mar 26 1997 12:35:59: 650PM"
```

After you load a transaction log using **until_time**, Adaptive Server restarts the database's log sequence. This means that until you dump the database again, you cannot load subsequent transaction logs after the **load transaction** using **until_time**. You will need to dump the database before you can dump another transaction log.

Bringing the Databases Online

In this example, the transaction log is loaded up to a time just before the table drop occurred. After you have applied all transaction log dumps to a database, use the **online database** command to make it available for use. In this example, the command to bring the *mydb* database online is:

```
online database mydb
```

Replicated Databases

Before upgrading replicated databases to release 11.5.x, the databases must be online. However, you cannot bring replicated databases online until the logs are drained. If you try to bring a replicated database online before the logs are drained, Adaptive Server issues the following message:

```
Database is replicated, but the log is not yet
drained. This database will come online
automatically after the log is drained.
```

When Replication Server, via the Log Transfer Manager (LTM), drains the log, the **online database** command is automatically issued.

Upgrading to Release 11.5.x

Refer to the the installation documentation for your platform for upgrade instructions for Adaptive Server users that have replicated databases.

Load Sequence

The load sequence for loading replicated databases is: **load database**, **replicate**, **load transaction**, **replicate**, and so on. At the end of the load sequence, issue the **online database** command to bring the databases online. Databases that are offline because they are in a load sequence are not automatically brought online by Replication Server. It is crucial that you do not issue the **online database** command until all transaction logs are loaded.

Upgrading User Database Dumps

When a SQL Server installation is upgraded to a new release, all databases associated with that server are automatically upgraded.

As a result, database and transaction log dumps created with any previous SQL Server must be upgraded before they can be used with Adaptive Server release 11.5.x.

Adaptive Server release 11.5.x provides an automatic upgrade mechanism, on a per-database basis, for upgrading a database or transaction log dump from any SQL Server release 10.0 or 11.0 to the current Adaptive Server release, thus making the dump compatible for use. This mechanism is entirely internal to Adaptive Server release 11.5.x and requires no external programs. It provides the flexibility of upgrading individual dumps as needed.

The following tasks are not supported by this automatic upgrade functionality:

- Loading a SQL Server release 10.0 *master* database onto Adaptive Server release 11.5.x.
- Installing new or modified stored procedures. Continue to use **installmaster.**
- Loading and upgrading dumps generated from SQL Server release 10.0.

How to Upgrade a Dump to Adaptive Server Release 11.5.x

To upgrade a user database or transaction log dump to the current release of Adaptive Server, follow these steps:

1. Use **load database** and **load transaction** to load the dump to be upgraded.

   Adaptive Server determines from the dump header which version it is loading. After the dump header is read, and before Backup Server begins the load, the database is marked offline by the **load database** or **load transaction** command. This makes the database unavailable for general use (queries and **use** *database* are not permitted), provides the user greater control over load sequences, and eliminates the possibility that other users will accidentally interrupt a load sequence.

2. Use the **online database** command, after the dump has successfully loaded, to activate the upgrade process.

   Do **not** issue the **online database** command until after all transaction dumps are loaded.

   Prior to SQL Server release 11.0, a database was automatically available at the end of a successful load sequence. With Adaptive Server release 11.5.x, the user is required to bring the database online after a successful load sequence, using the **online database** command.

   For dumps loaded from SQL Server release 10.0, the **online database** command activates the upgrade process to upgrade the dumps just loaded. After the upgrade is successfully completed, Adaptive Server places the database online and the database is ready for use.

   For dumps loaded from Adaptive Server release 11.5.x, no upgrade process is activated. You must still issue the **online database** command to place the database online, making it available for public use. (The **load database** command marks the database offline.)

   During the upgrade process, each upgrade step produces a message stating what it is about to do.

   An upgrade failure leaves the database offline and produces a message stating that the upgrade failed and the user must correct the failure.

   For more information about **online database**, see the *Adaptive Server Reference Manual*.

3. After successful execution of the **online database** command, use the **dump database** command. The database must be dumped before a **dump transaction** is permitted. A **dump transaction** on a newly created or upgraded database is not permitted until a successful **dump database** has occurred.

The "Database Offline" Status Bit

The "database offline" status bit indicates that the database is not available for general use. You can determine whether a database is offline by using **sp_helpdb**. It will show that the database is offline if this bit is set.

When a database is marked offline by **load database**, a status bit in the *sysdatabases* table is set and remains set until the successful completion of **online database.**

The "database offline" status bit works in combination with any existing status bits. It augments the following status bit to provide additional control:

- In recovery

The "database offline" status bit overrides the following status bits:

- DBO use only
- Read only

The following status bits override the "database offline" status bit:

- Began upgrade
- Bypass recovery
- In load
- Not recovered
- Suspect
- Use not recovered

Although the database is not available for general use, the following commands are permitted when the database is offline:

- **dump database** and **dump transaction**
- **load database** and **load transaction**
- **alter database on device**
- **drop database**
- **online database**

- **dbcc** diagnostics (subject to **dbcc** restrictions)

Version Identifiers

The automatic upgrade feature provides version identifiers for Adaptive Server, databases, and log record formats. They are:

Configuration Upgrade Version ID

This identifier shows the current version of Adaptive Server; it is stored in the *sysconfigures* system table. **sp_configure** displays the current version of Adaptive Server as "upgrade version."

Upgrade Version Indicator

This identifier shows the current version of a database and is stored in the database and dump headers. The Adaptive Server recovery mechanism uses this value to determine whether the database should be upgraded before being made available for general use.

Log Compatibility Version Specifier

This identifier differentiates release 10.x logs from release 11.x logs by showing the format of log records in a database, database dump, or transaction log dump. This constant is stored in the database and dump headers and is used by Adaptive Server to detect the format of log records during recovery.

Cache Bindings and Loading Databases

You need to be aware of cache bindings for a database and the objects in the database if you dump a database and load it onto a server with different cache bindings. You may want to load the database onto a different server for tuning or development work, or you may need to load a database that you dropped from a server whose cache bindings have changed since the dump was made.

When a database is brought online after recovery or with the **online database** command after a load, Adaptive Server verifies all cache bindings for the database and database objects. If a cache does not exist, Adaptive Server writes a warning to the error log, and the binding in *sysattributes* is marked as invalid. Here is an example of the message from the errorlog:

```
Cache binding for database '5', object
'208003772', index '3' is being marked invalid in
Sysattributes.
```

Invalid cache bindings are not deleted. If you create a cache of the same name and restart Adaptive Server, the binding is marked as valid and the cache is used. If you do not create a cache with the same name, you can bind the object to another cache or allow it to use the default cache.

In the following sections, which discuss cache binding topics, **destination server** refers to the server where the database is being loaded, and **original server** refers to the server where the dump was made.

If possible, re-create caches that have the same names on the destination server as the bindings on the original server. You may want to configure pools in exactly the same manner if you are using the destination database for similar purposes or for performance testing and development that may be ported back to the original server. If you are using the destination database for decision support or for running **dbcc** commands, you may want to configure pools to allow more space in 16K memory pools.

Databases and Cache Bindings

Binding information for databases is stored in *master..sysattributes*. No information about database binding is stored in the database itself. If you use **load database** to load the dump over an existing database that is bound to a cache, and you do not drop the database before you issue the load command, this does not affect the binding.

If the database that you are loading was bound to a cache on the original server, your options are:

- Bind the database on the destination server to a cache configured for the needs on that server, or
- Configure pools in the default data cache on the destination server for the needs of the application there, and do not bind the database to a named data cache.

Database Objects and Cache Bindings

Binding information for objects is stored in the *sysattributes* table in the database itself. If you frequently load the database onto the

destination server, the simplest solution is to configure caches of the same name on the destination server.

If the destination server is not configured with caches of the same name as the original server, bind the objects to the appropriate caches on the destination server after you bring the database online, or be sure that the default cache is configured for your needs on that server.

Checking on Cache Bindings

The **sp_helpcache** system procedure displays the cache bindings for database objects, even if the cache bindings are invalid.

The following SQL statements reproduce cache binding commands from the information in a user database's *sysattributes* table:

```
/* create a bindcache statement for tables */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", " + object_name(object)
from sysattributes
where class = 3
    and object_type = "T"

/* create a bindcache statement for indexes */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", "   + i.name
from sysattributes, sysindexes i
where class = 3
    and object_type = "I"
    and i.indid = convert(tinyint, object_info1)
    and i.id = object
```

Cross-Database Constraints and Loading Databases

If you use the **references** constraint of **create table** or **alter table** to reference tables across databases, you may encounter problems when you try to load a dump of one of these databases.

- If tables in a database reference a dumped database, referential integrity errors result if you load the database with a different name or on a different server from where it was dumped. If you want to change the name or location of a database when you reload it, use **alter table** in the referencing database to drop all

external referential integrity restraints before you dump the database.

- Loading a dump of a referenced database that is earlier than the referencing database could cause consistency issues or data corruption. As a precaution, each time you add or remove a cross-database constraint or drop a table that contains a cross-database constraint, dump both affected databases.
- Dump all databases that reference each other at the same time. To guard against synchronization problems, put both databases in single-user mode for the dumps. When loading the databases, bring both databases online at the same time.

Cross-database constraints can become inconsistent if you:

- Do not load database dumps in chronological order (for example, you load a dump created on August 12, 1997, after one created on August 13), or
- Load a dump into a database with a new name.

If you do not load , cross-database constraints can become inconsistent.

To remedy this problem, follow these steps:

1. Put both databases in single user mode.
2. Drop the inconsistent referential constraint.
3. Check the data consistency with a query such as the following:

   ```
   select foreign_key_col from table
   where foreign_key not in
   (select primary_key_col from other db..othertable)
   ```

4. Fix any data inconsistency problems.
5. Re-create the constraint.

22 Restoring the System Databases

This chapter is the third of a four-chapter unit on backup and recovery. It explains how to restore the *master, model* and *sybsystemprocs* databases. Topics include:

- What Does Recovering a System Database Entail? 22-1
- Symptoms of a Damaged master Database 22-2
- Recovering the master Database 22-2
- Recovering the model Database 22-17
- Recovering the sybsystemprocs Database 22-18
- Restoring System Tables with disk reinit and disk refit 22-21

Table 22-1 lists additional sources of information about backup and recovery:

Table 22-1: Further information about backup and recovery

For More Information About	See
Backup and recovery issues to address before production	Chapter 20, "Developing a Backup and Recovery Plan"
dump, **load**, and **sp_volchanged** syntax	Chapter 21, "Backing Up and Restoring User Databases"
Using thresholds to automate backups	Chapter 23, "Managing Free Space with Thresholds"

What Does Recovering a System Database Entail?

The recovery procedure for system databases depends on the database involved and the problems that you have on your system. In general, the recovery may include:

- Using **load database** to load backups of these databases,
- Using **buildmaster, installmaster** and **installmodel** to restore the initial state of these databases, or
- A combination of the above tasks.

Follow the instructions in this chapter carefully to understand the correct recovery procedure to use.

To make the recovery of system databases as efficient as possible, follow these good system administration practices:

- Do not store user databases or any databases other than *master*, *tempdb*, and *model* on the master device.
- Always keep up-to-date printouts of important system tables.
- Always back up the *master* database after performing actions such as initializing database devices, creating or altering databases, or adding new server logins.

The purpose for these recommendations is to simplify recovering from failure of the *master* database or master device. See also "master Database" on page 2-2 and "Scheduling Routine Backups" on page 20-34 for more information on using good administration practices.

Symptoms of a Damaged *master* Database

A damaged *master* database can be caused by a media failure in the area on which *master* is stored or by internal corruption in the database. A damaged *master* database makes itself known in one or more of these ways:

- Adaptive Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- **dbcc** (the database consistency checker) reports damage during a regularly scheduled check of your databases.

Recovering the *master* Database

This section describes the steps to recover the *master* database and to rebuild the master device. It assumes the following:

- The *master* database is corrupt, or the master device is damaged
- You have up-to-date printouts of the important system tables, as listed in "Backing Up master and Keeping Copies of System Tables" on page 2-4
- The master device contains only the *master* database, *tempdb*, and *model*, and no other Sybase-supplied or user databases
- You have an up-to-date backup of the *master* database, and you have not initialized any devices or created or altered any databases since last dumping *master*.

- Your server uses the default sort order.

You can also use these procedures to move your *master* database to a larger master device.

This chapter does not cover more complicated disaster recovery scenarios. For more complicated scenarios, you will have to perform additional steps. The *Troubleshooting Guide* provides more complete coverage of recovery scenarios.

About the Recovery Process

Special procedures are needed because of the central, controlling nature of the *master* database and the master device. Tables in *master* configure and control all Adaptive Server's functions, databases, and data devices. The recovery process:

- Rebuilds the master device to its default state when you first installed a server
- Restores the *master* database to the default state
- Restores the *master* database to its condition at the time of your last backup

During the early stages of recovering the *master* database, you will not be able to use the system stored procedures.

Summary of Recovery Procedure

System Administrators must follow the steps below to restore a damaged master device. **Each step is discussed in more detail on the following pages.**

Step	See
Find hard copies of the system tables needed to restore disks, databases and logins.	"Step One: Find Copies of System Tables" on page 22-4
Shut down Adaptive Server, and use **buildmaster** to build a new **master** database and master device.	"Step Two: Build a New Master Device" on page 22-5
Restart Adaptive Server in master-recover mode.	"Step Three: Start Adaptive Server in Master-Recover Mode" on page 22-6

Step	See
Re-create the *master* database's allocations in *sysusages* exactly.	"Step Four: Re-Create Device Allocations for master" on page 22-7
Update Backup Server's network name in the *sysservers* table.	"Step Five: Check Your Backup Server sysservers Information" on page 22-12
Verify that your Backup Server is running.	"Step Six: Verify That Your Backup Server Is Running" on page 22-13
Use **load database** to load the most recent database dump of *master*. Adaptive Server stops automatically after successfully loading *master*.	"Step Seven: Load a Backup of master" on page 22-13
Update the **number of devices** configuration parameter in the configuration file.	"Step Eight: Update the number of devices Configuration Parameter" on page 22-13.
Restart Adaptive Server in single-user mode.	"Step Nine: Restart Adaptive Server in Master-Recover Mode" on page 22-14
Verify that the backup of *master* has the latest system tables information.	"Step Ten: Check System Tables to Verify Current Backup of master" on page 22-14
Restart Adaptive Server.	"Step Eleven: Restart Adaptive Server" on page 22-15
Check *syslogins* if you have added new logins since the last backup of *master*.	"Step Twelve: Restore Server User IDs" on page 22-15
Restore the *model* database.	"Step Thirteen: Restore the model Database" on page 22-15
Check for consistency: compare your hard copies of *sysusages* and *sysdatabases* with the new online version, run **dbcc checkalloc** on each database, and examine the important tables in each database.	"Step Fourteen: Check Adaptive Server" on page 22-16
Dump the *master* database.	"Step Fifteen: Back Up master" on page 22-16

The following sections describe these steps in detail.

Step One: Find Copies of System Tables

Find copies of the following system tables that you have saved to a file: *sysdatabases*, *sysdevices*, *sysusages*, *sysloginroles* and *syslogins*. You

can use these to guarantee that your system has been fully restored at the completion of this process.

For information on preparing for disaster recovery by making copies of the system tables to a file, see "Backing Up master and Keeping Copies of System Tables" on page 2-4.

Step Two: Build a New Master Device

The next step is to shut down Adaptive Server, if it is running, and to rebuild the master device. When rebuilding the master device, you must specify the device size.

Before you begin, it is important that you remember the following information:

- Use a new device, preserving the old device in case you encounter problems. The old device may provide crucial information.
- Shut down Adaptive Server before you use any **buildmaster** command. If you use **buildmaster** on a master device that is in use by Adaptive Server, the recovery procedure will fail when you attempt to load the most recent backup of *master*.

Run **buildmaster** (UNIX), **bldmastr** (Windows NT) or **buildmaster** (OpenVMS) to build a new master device and to install a copy of a "generic" *master* database. Give the full name for your master device, and the full size of the device.

➤ ***Note***

You must give **buildmaster** a size as large as or larger than the size originally used to configure Adaptive Server. If the size you provide is too small, you will get error messages when you try to load your databases.

This example rebuilds a 17MB (8704 2K pages) master device:

On UNIX platforms:

```
buildmaster -d /dev/rsd1f -s8704
```

On Windows NT:

```
bldmastr -d d:\devices\master.dat -s870
```

On OpenVMS:

```
buildmaster
/disk=dua0:[devices.master]d_master.dat/size=8704
```

After you run **buildmaster**, the password for the default "sa" account reverts to NULL.

For details on the **buildmaster** utility, see the the *Utility Programs* manual for your platform manual.

Step Three: Start Adaptive Server in Master-Recover Mode

Start Adaptive Server in master-recover mode with the **-m** (UNIX and Windows NT) or **/masterrecover** (OpenVMS) options.

On UNIX platforms, make a copy of the runserver file, naming it *m_RUN_server_name*. Edit the new file, adding the parameter **-m** to the **dataserver** command line. Then start the server in master-recover mode:

```
startserver -f m_RUN_server_name
```

On OpenVMS, use the following command:

```
startserver /server = server_name /masterrecover
```

On Windows NT, start Adaptive Server from the command line using the **sqlsrver** command, specifying the **-m** parameter in addition to other necessary parameters. For example:

```
sqlsrver.exe -dD:\Sybase\DATA\MASTER.dat -sPIANO -
eD:\Sybase\install\errorlog -iD:\Sybase\ini -MD:\Sybase -m
```

See the *Utility Programs* manual for your platform manual for the complete syntax of these commands.

When you start Adaptive Server in master-recover mode, only one login of one user—the System Administrator—is allowed. Immediately following a **buildmaster** command on the *master* database, only the "sa" account exists, and its password is NULL.

◆ ***WARNING!***

Some sites have automatic jobs that log into the server at start-up with the "sa" login. Be sure these are disabled.

The special master-recover mode is necessary because the generic *master* database created with **buildmaster** does not match the actual situation in Adaptive Server. For example, the database does not know about any of your database devices. Any operations on the *master* database could make recovery impossible or at least much more complicated and time-consuming.

An Adaptive Server started in master-recover mode is automatically configured to allow direct updates to the system tables. Certain other operations (for example, the checkpoint process) are disallowed.

◆ ***WARNING!***

Ad hoc changes to system tables are dangerous—some changes can render Adaptive Server unable to run. Make only the changes described in this chapter, and always make the changes in a user-defined transaction.

Step Four: Re-Create Device Allocations for *master*

Before you run **buildmaster**, check your most recent copy of *sysusages*. If it has only one line for *dbid* 1, your *master* database has only one disk allocation piece, and you can go to "Step Five: Check Your Backup Server sysservers Information" on page 22-12.

If more than one row for *dbid* 1 appears in your hard copy of *sysusages*, you need to increase the size of *master* so that you can load the dump. You must duplicate the *vstart* value for each allocation for *master* in *sysusages.* This is easiest to do if you have a copy of *sysusages* ordered by *vstart*.

In the simplest cases, additional allocations to *master* only require the use of **alter database.** In more complicated situations, you must allocate space for other databases in order to reconstruct the exact *vstart* values needed to recover *master.*

In addition to the *master* database, *tempdb* (*dbid* =2) and *model* (*dbid* = 3) are located wholly or partially on the master device.

Determining Which Allocations Are on the Master Device

To determine which *vstart* values represent allocations on the master device, look at your hard copy of the *sysdevices* table. It shows the low and high values for each device. Databases devices always have a *cntrltype* of 0; the rows for tape devices are not included in this sample.

```
low      high     status cntrltype name     phyname  mirrorname
---      ----     ------ --------- ------   -------- ----------
  0      8703     3              0 master   d_master NULL
16777216 16782335 2              0 sprocdev /sybase/ NULL
                                            sp_dev
```

page range for master device

Figure 22-1: Determining allocations on the master device

In this example, page numbers on the master device are between 0 and 8703, so any database allocation with *sysusages.vstart* values in this range represent allocations on the master device.

Check all rows for *master* (except the first) in your saved *sysusages* output. Here is sample *sysusages* information, ordered by *vstart*:

```
dbid segmap lstart size   vstart     pad  unreservedpgs
---- ------ ------ ------ ---------- ---- -------------
   1      7      0   1536          4 NULL           480
   3      7      0   1024       1540 NULL           680
   2      7      0   1024       2564 NULL           680
   1      7   1536   1024       3588 NULL          1024
   4      7      0   5120       4432 NULL          1560
```

dbid = 1, an additional allocation for master

size of this allocation

vstart between 0 and 8703

Figure 22-2: Sample output from sysusages

In this example, the first four rows have *vstart* values between 4 and 3588. Only *dbid* 4 is on another device.

buildmaster re-creates the first three rows, so *sysusages* in your newly rebuilt *master* database should match your hard copy.

The fourth row shows an additional allocation for *master* with *vstart* = 3588 and *size* = 1024.

Figure 22-3 shows the storage allocations for the above *sysusages* data.

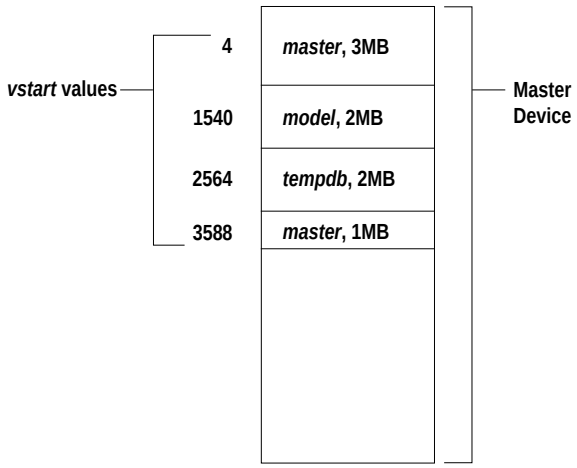


Figure 22-3: Allocations on a master device

In Figure 22-3, you only need to issue an **alter database** command to increase the size of the *master* database. To determine the size to provide for this command, look at the *size* column for the second allocation to *master.* Divide by 512. In this example, the additional row for *master* indicates an allocation of 1024 data pages, so the correct parameter is 2, the result of 1024 / 512.

Use that result for the **alter database** command. Log into the server as "sa". Remember that **buildmaster** has set the password for this account to NULL. Issue the **alter database** command. For the example above, use:

```
alter database master on master = 2
```

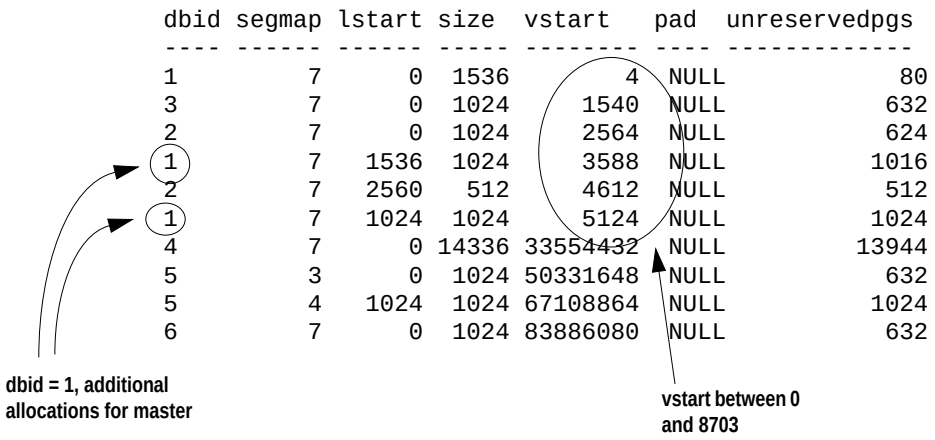
Check the *size* and *vstart* values for the new row in *sysusages*.

Creating Additional Allocations

Your output from *sysusages* may have more allocations on the master device if:

- Adaptive Server has been upgraded from an earlier SQL Server release
- A System Administrator has increased the size of *master, model,* or *tempdb* on the master device

You must restore these allocations up to the last row for the *master* database, *dbid* 1. Here is an example of *sysusages* showing additional allocations on the master device, in order by *vstart*:



dbid	segmap	lstart	size	vstart	pad	unreservedpgs
1	7	0	1536	4	NULL	80
3	7	0	1024	1540	NULL	632
2	7	0	1024	2564	NULL	624
1	7	1536	1024	3588	NULL	1016
2	7	2560	512	4612	NULL	512
1	7	1024	1024	5124	NULL	1024
4	7	0	14336	33554432	NULL	13944
5	3	0	1024	50331648	NULL	632
5	4	1024	1024	67108864	NULL	1024
6	7	0	1024	83886080	NULL	632

Figure 22-4: Sample sysusages output with additional allocations

This copy of *sysusages* shows the following allocations on the master device (excluding the three created by **buildmaster**):

- One for *master*, *dbid* = 1, *size* = 1024, *vstart* = 3588
- One for *tempdb*, *dbid* = 2, *size* = 512, *vstart* = 4612
- Another allocation for *master*, *dbid* = 1, *size* = 1024, *vstart* = 5124

The final allocations in this output are not on the master device.

Figure 22-5 shows the allocations on the master device.

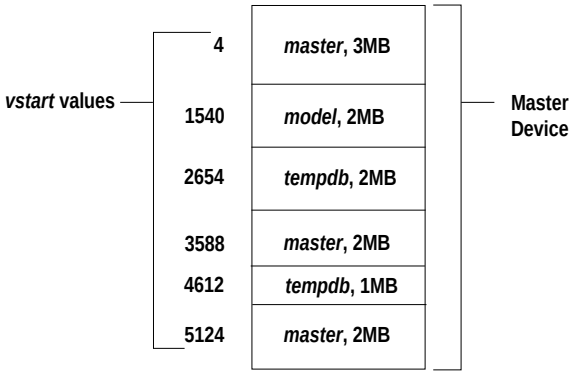


Figure 22-5: Complex allocations on a master device

You need to issue a set of **alter database** and **create database** commands to re-create all the allocations with the correct sizes and *vstart* values. If your *sysusages* table lists additional allocations on the master device after the last allocation for master, you do not have to re-create them.

To determine the size for the **create database** and **alter database** commands, divide the value shown in the *size* column of the *sysusages* output by 512.

To reconstruct the allocation, issue these commands, in this order:

- To restore the first allocation to *master, dbid* 1, *size* = 1024:

  ```
  alter database master on default = 2
  ```

- To allocate more space to *tempdb, dbid* 2, *size* = 512:

  ```
  alter database tempdb on default = 1
  ```

- To add the final allocation to *master, dbid* 1, *size* = 1024:

  ```
  alter database master on default = 1
  ```

You need to restore only the allocations up to and including the last line for the *master* database. When you load the backup of *master,* this table is completely restored from the dump.

At this point, carefully check the current *sysusages* values with the values in your hard copy:

- If all of the *vstart* and *size* values for *master* match, go to “Step Five: Check Your Backup Server sysservers Information” on page 22-12.

- If the values do not match, an attempt to load the *master* database will almost certainly fail. Shut down the server, and begin again by running **buildmaster**. See “Step Two: Build a New Master Device” on page 22-5.
- If your *sysusages* values look correct, go to “Step Five: Check Your Backup Server sysservers Information”.

Step Five: Check Your Backup Server *sysservers* Information

Log into the server as “sa”, using a null password.

If the network name of your Backup Server is not SYB_BACKUP, you must update *sysservers* so that Adaptive Server can communicate with its Backup Server. Check the Backup Server name in your interfaces file, and issue this command:

```
select *
from sysservers
where srvname = "SYB_BACKUP"
```

Check the *srvnetname* in the output from this command. If it matches the interfaces file entry for the Backup Server for your server, go to “Step Six: Verify That Your Backup Server Is Running” on page 22-13.

If the *srvnetname* reported by this command is **not** the same as the name of your Backup Server in the interfaces file, you must update *sysservers*. The example below changes the Backup Server’s network name to PRODUCTION_BSRV:

```
begin transaction

update sysservers
set srvnetname = "PRODUCTION_BSRV"
where srvname = "SYB_BACKUP"
```

Execute this command, and check to be sure that it modified only one row. Issue the **select** command again, and check to be sure that the correct row was modified and that it contains the correct value. If the **update** command modified more than one row, or if it modified the wrong row, issue a **rollback transaction** command, and attempt the update again.

If the command correctly modified the Backup Server’s row, issue a **commit transaction** command.

Step Six: Verify That Your Backup Server Is Running

On UNIX and OpenVMS platforms, use the **showserver** command to verify that your Backup Server is running; restart your Backup Server if necessary. See **showserver** and **startserver** in the *Utility Programs* manual for your platform manual.

On Windows NT, a locally installed Sybase Central and the Services Manager utilities show whether Backup Server is running.

See the *Utility Programs* manual for your platform manual for the commands to start Backup Server.

Step Seven: Load a Backup of *master*

Load the most recent backup of the *master* database with **load database**. Here are examples of the load commands:

On UNIX platforms:

```
load database master from "/dev/nrmt4"
```

On OpenVMS:

```
load database master from "MTA0:"
```

On Windows NT:

```
load database master from "\\.\TAPE0"
```

See Chapter 21, “Backing Up and Restoring User Databases,” for information on command syntax.

After the **load database** command completes successfully, Adaptive Server shuts itself down. Watch for any error messages during the load and shut down processes.

Step Eight: Update the *number of devices* Configuration Parameter

Perform this step only if you use more than the default number of database devices. Otherwise, go to “Step Nine: Restart Adaptive Server in Master-Recover Mode” on page 22-14.

Configuration values are not available to Adaptive Server until after recovery of the *master* database, so you need instruct Adaptive Server to read the appropriate value for the **number of devices** parameter from a configuration file at start-up.

Edit a configuration file to reflect the correct value for the **number of devices** parameter, if your most recent configuration file is not available.

Edit the runserver file. Add the **-c** parameter to the end of the **dataserver** or **sqlsrver** command, specifying the name and location of the configuration file. When Adaptive Server starts, it will read in the parameter values from the specified configuration file.

Step Nine: Restart Adaptive Server in Master-Recover Mode

Use **startserver** to restart Adaptive Server in master-recover mode (see "Step Three: Start Adaptive Server in Master-Recover Mode" on page 22-6). Watch for error messages during recovery.

Loading the backup of *master* restores the "sa" account to its previous state. It restores the password on the "sa" account, if one exists. If you used **sp_locklogin** to lock this account before the backup was made, the "sa" account will now be locked. Perform the rest of these steps using an account with the System Administrator role.

Step Ten: Check System Tables to Verify Current Backup of *master*

If the backup of your *master* database is up to date—that is, if you have backed up the *master* database since issuing the most recent **disk init**, **create database**, or **alter database** command—then the contents of *sysusages, sysdatabases*, and *sysdevices* will match your hard copy.

Check the *sysusages, sysdatabases* and *sysdevices* tables in your recovered server against your hard copy. Look especially for these problems:

- If any devices in your hard copy are not included in the restored *sysdevices*, then you have added devices since your last backup, and you must run **disk reinit** and **disk refit**. For information on using these commands, see "Restoring System Tables with disk reinit and disk refit" on page 22-21.
- If any databases listed in your hard copy are not listed in your restored *sysdatabases* table, you have added a database since the last time you backed up *master*. You must run **disk refit** (see "Restoring System Tables with disk reinit and disk refit" on page 22-21).

Step Eleven: Restart Adaptive Server

Restart Adaptive Server in normal (multiuser) mode.

Step Twelve: Restore Server User IDs

Check your hard copy of *syslogins* and your restored *syslogins* table. Look especially for the following situations and reissue the appropriate commands, as necessary:

- If you have added server logins since the last backup of *master*, reissue the **sp_addlogin** commands.
- If you have dropped server logins, reissue the **sp_droplogin** commands.
- If you have locked server accounts, reissue the **sp_locklogin** commands.
- Check for other differences caused by the use of the **sp_modifylogin** procedure by users or by System Administrators.

Make sure that the *suids* assigned to users are correct. Mismatched *suid* values in databases can lead to permission problems, and users may not be able to access tables or run commands.

An effective technique for checking existing *suid* values is to perform a **union** on each *sysusers* table in your user databases. You can include *master* in this procedure, if users have permission to use *master*.

For example:

```
select suid, name from master..sysusers
union
select suid, name from sales..sysusers
union
select suid, name from parts..sysusers
union
select suid, name from accounting..sysusers
```

If your resulting list shows skipped *suid* values in the range where you need to redo the logins, you must add placeholders for the skipped values and then drop them with **sp_droplogin** or lock them with **sp_locklogin.**

Step Thirteen: Restore the *model* Database

Restore the *model* database:

- Load your backup of *model*, if you keep a backup.
- If you do not have a backup:
  - Run the **installmodel** script:

    On most platforms:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < installmodel
```

    On Windows NT:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < instmodl
```

    On OpenVMS:

```
set default sybase_system:[sybase.scripts]
define dsquery server_name
isql/user="sa"/password="password"
 /input=installmodel
```

  - Redo any changes you made to *model*.

Step Fourteen: Check Adaptive Server

Check Adaptive Server carefully:

1. Compare your hard copy of *sysusages* with the new online version.
2. Compare your hard copy of *sysdatabases* with the new online version.
3. Run **dbcc checkalloc** on each database.
4. Examine the important tables in each database.

◆ ***WARNING!***

If you find discrepancies in *sysusages*, call Sybase Technical Support.

Step Fifteen: Back Up *master*

When you have completely restored the *master* database and have run full **dbcc** integrity checks, back up the database using your usual dump commands.

Recovering the *model* Database

This section describes recovery of the *model* database when only the *model* database needed to be restored. It includes instructions for these scenarios:

- You have not made any changes to *model*, so you need to restore only the generic *model* database.
- You have changed *model*, and you have a backup.
- You have changed *model*, and you do not have a backup.

Restoring the Generic *model* Database

buildmaster can restore the *model* database without affecting *master*.

◆ ***WARNING!***

Shut down Adaptive Server before you use any buildmaster command.

On UNIX platforms:

```
buildmaster -d /devname -x
```

On OpenVMS:

```
buildmaster /disk = physicalname /model
```

On Windows NT:

```
bldmastr -d physicalname -x
```

Restoring *model* from a Backup

If you can issue the **use model** command successfully, you can restore your *model* database from a backup with the **load database** command.

If you cannot use the database:

1. Follow the instructions above for “Restoring the Generic model Database” on page 22-17.
2. If you have changed the size of *model*, reissue the **alter database** command.
3. Load the backup with **load database**.

Restoring *model* with No Backup

If you have changed your *model* database, and you do not have a backup:

- Follow the steps for "Restoring the Generic model Database" on page 22-17.
- Reissue all the commands you issued to change *model*.

Recovering the *sybsystemprocs* Database

The *sybsystemprocs* database stores the system procedures that are used to modify and report on system tables. If your routine **dbcc** checks on this database report damage, and you do not keep a backup of this database, you can restore it using **installmaster**. If you do keep backups of *sybsystemprocs*, you can restore it with **load database.**

Restoring *sybsystemprocs* with *installmaster*

1. Check to see what logical device currently stores the database. If you can still use **sp_helpdb**, issue this command:

   sp_helpdb sybsystemprocs

```
name                 db_size       owner            dbid
        created
        status
------------------- ------------- ---------------- ------
sybsystemprocs            28.0 MB sa                    4
        Aug 07, 1993
        trunc log on chkpt

device_fragments   size        usage            free kbytes
------------------ ----------- ---------------- -----------
sprocdev           28.0 MB     data and log            3120
```

   The "device_fragments" column indicates that the database is stored on *sprocdev*.

   If you cannot use **sp_helpdb**, this query reports the devices used by the database and the amount of space on each device:

```
select sysdevices.name, sysusages.size / 512
from sysdevices, sysdatabases, sysusages
where sysdatabases.name = "sybsystemprocs"
  and sysdatabases.dbid = sysusages.dbid
  and sysdevices.low <= sysusages.size + vstart
  and sysdevices.high >= sysusages.size + vstart -1
name
--------------- -------
sprocdev              28
```

2. Drop the database:

   ```
   drop database sybsystemprocs
   ```

   If the physical disk is damaged, use **dbcc dbrepair** to drop the database and then use **sp_dropdevice** to drop the device. If necessary, use **disk init** to initialize a new database device. See Chapter 6, "Initializing Database Devices," for more information on **disk init**.

3. Re-create the *sybsystemprocs* database on the device, using the size returned by the query under Step 1:

   ```
   create database sybsystemprocs
       on sprocdev = 28
   ```

➤ ***Note***

The required size for the *sybsystemprocs* database may be different for your operating system. See the installation documentation for your platform for the correct size.

4. Run the **installmaster** script.

◆ ***WARNING!***

Do not run the *installmaster* script repeatedly without first dropping and re-creating the *sybsystemprocs* database. Running *installmaster* repeatedly can change the distribution of index values in such a way that the *sysprocedures* table will require much more disk space to store the same amount of data. To avoid this problem, drop and re-create the *sybsystemprocs* database before running *installmaster*.

On UNIX platforms:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < installmaster
```

On OpenVMS:

```
set default sybase_system:[sybase.scripts]
define dsquery server_name
isql/user="sa"/password="password"
     /input=installmaster
```

On Windows NT:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < instmstr
```

5. If you have made any changes to permissions in *sybsystemprocs*, or if you have added your own procedures to the database, you must redo the changes.

Restoring *sybsystemprocs* with *load database*

If you write system procedures and store them in the *sybsystemprocs* database, you have two ways to recover them if the database is damaged:

- Restore the database from **installmaster**, as described in Step 4 under "Restoring sybsystemprocs with installmaster" on page 22-18. Then re-create the procedures by reissuing the **create procedure** commands.
- Keep backups of the database, and load them with **load database.**

If you choose to keep a backup of the database, be sure that the complete backup fits on one tape volume or that more than one Adaptive Server is able to communicate with your Backup Server. If a dump spans more than one tape volume, issue the change-of-volume command using the system procedure **sp_volchanged**, which is stored in *sybsystemprocs*. You cannot issue that command in the middle of recovering a database.

Following are sample load commands:

On UNIX:

```
load database sybsystemprocs from "/dev/nrmt4"
```

On OpenVMS:

```
load database sybsytemprocs from "MTA0:"
```

On Windows NT:

```
load database sybsystemprocs from "\\.\TAPE0"
```

Restoring System Tables with *disk reinit* and *disk refit*

In situations in which you are restoring the *master* database from a dump that does not reflect the most recent **disk init** or **create database** and **alter database** commands, follow procedures listed in this section for restoring the proper information in the *sysusages, sysdatabases* and *sysdevices* tables.

Check the *sysusages*, *sysdatabases*, and *sysdevices* tables in your recovered server against your hard copy. Look especially for these problems:

- If any devices in your hard copy are not included in the restored *sysdevices,* then you have added devices since your last backup, and you must run **disk reinit** and **disk refit.**
- If any databases listed in your hard copy are not listed in your restored *sysdatabases* table, then you have added a database since the last time you backed up *master.* You must run **disk refit.**

Restoring *sysdevices* with *disk reinit*

If you have added any database devices since the last dump—that is, if you have issued a **disk init** command—you must add each new device to *sysdevices* with the **disk reinit** command. If you saved scripts from your original **disk init** commands, use them to determine the parameters for the **disk reinit** command (including the original value of *vstart*). If the size you provide is too small, or if you use a different *vstart* value, you may corrupt your database.

If you do not save your **disk init** scripts, look at your most recent hard copy of *sysdevices* to determine some of the correct parameters for **disk reinit.** You will still need to know the value of *vstart* if you used a custom *vstart* in the original **disk init** command.

Table 22-2 describes the **disk reinit** parameters and their corresponding sysdevices data:

Table 22-2: Using sysdevices to determine disk reinit parameters

disk reinit Parameter	*sysdevices* Data	Notes
name	*name*	Use the same name, especially if you have any scripts that create or alter databases or add segments.
physname	*phyname*	Must be full path to device.

Table 22-2: Using sysdevices to determine disk reinit parameters (continued)

disk reinit Parameter	*sysdevices* Data	Notes
vdevno	*low* / 16777216	Not necessary to use the same value for *vdevno*, but be sure to use a value not already in use.
size	(*high*-*low*) +1	Extremely important to provide correct size information.

You can also obtain information on devices by reading the errorlog for *name*, *physname*, and *vdevno*, and using operating system commands to determine the size of the devices.

If you store your *sybsystemprocs* database on a separate physical device, be sure to include a **disk reinit** command for *sybsystemprocs*, if it is not listed in *sysdevices*.

After running **disk reinit**, compare your *sysdevices* table to the copy you made before running **buildmaster**.

disk reinit can be run only from the *master* database and only by a System Administrator. Permission cannot be transferred to other users. Its syntax is:

```
disk reinit
      name = "device_name",
      physname = "physical_name",
      vdevno = virtual_device_number,
      size = number_of_blocks
      [, vstart = virtual_address,
      cntrltype = controller_number]
```

For more information on **disk reinit**, see the discussion of **disk init** in Chapter 6, “Initializing Database Devices,” or see the *Adaptive Server Reference Manual*.

Restoring *sysusages* and *sysdatabase* with *disk refit*

If you have added database devices or created or altered databases since the last database dump, use **disk refit** to rebuild the *sysusages* and *sysdatabases* tables.

disk refit can be run only from the *master* database and only by a System Administrator. Permission cannot be transferred to other users. Its syntax is:

```
disk refit
```

Adaptive Server shuts down after **disk refit** rebuilds the system tables. Examine the output while **disk refit** runs and during the shutdown process to determine whether any errors occurred.

◆ ***WARNING!***

Providing inaccurate information in the disk reinit command may lead to permanent corruption when you update your data. Be sure to check Adaptive Server with dbcc after running disk refit.

23 Managing Free Space with Thresholds

When you create or alter a database, you allocate a finite amount of space for its data and log segments. As you create objects and insert data, the amount of free space in the database decreases.

This chapter is the last of a four-chapter unit on backup and recovery. It explains how to use thresholds to monitor the amount of free space in a database segment. Topics include:

- Monitoring Free Space with the Last-Chance Threshold 23-1
- Aborting or Suspending Processes 23-4
- Waking Suspended Processes 23-4
- Adding, Changing, and Deleting Thresholds 23-5
- Creating a Free-Space Threshold for the Log Segment 23-8
- Creating Additional Thresholds on Other Segments 23-11
- Creating Threshold Procedures 23-12
- Disabling Free-Space Accounting for Data Segments 23-18

Table 23-1 lists additional sources of information about backup and recovery:

Table 23-1: Further information about backup and recovery

For More Information About	See
Backup and recovery issues to address before production	Chapter 20, "Developing a Backup and Recovery Plan"
dump, **load**, and **sp_volchanged** syntax	Chapter 21, "Backing Up and Restoring User Databases"
Backing up and restoring the system databases	Chapter 22, "Restoring the System Databases"

Monitoring Free Space with the Last-Chance Threshold

A threshold always has a stored procedure associated with it, and the threshold acts like a trip wire. When free space on the segment falls below the amount specified by the threshold, Adaptive Server executes the stored procedure.

Each database that stores its transaction log on a separate segment has a **last-chance threshold**. The threshold is an estimate of the number of free log pages that would be required to back up the transaction log. Adaptive Server automatically adjusts the last-chance threshold as you allocate more space to the log segment.

When the amount of free space in the log segment falls below the last-chance threshold, Adaptive Server automatically executes a special stored procedure called **sp_thresholdaction.** (You can specify a different last-chance threshold procedure with **sp_modifythreshold**.)

Figure 23-1 illustrates a log segment with a last-chance threshold. The shaded area represents log space that has already been used; the unshaded area represents free log space. The last-chance threshold has not yet been crossed.

Figure 23-1: Log segment with a last-chance threshold

Crossing the Threshold

As users execute transactions, the amount of free log space decreases. When the amount of free space crosses the last-chance threshold, Adaptive Server executes **sp_thresholdaction**:

Figure 23-2: Executing sp_thresholdaction when the last-chance threshold is reached

Controlling How Often *sp_thresholdaction* Executes

Adaptive Server uses a **hysteresis value**, the global variable *@@thresh_hysteresis*, to control how sensitive thresholds are to variations in free space. Once a threshold executes its procedure, it is deactivated. The threshold remains inactive until the amount of free space in the segment rises *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space.

For example, when the threshold in Figure 23-2 executes **sp_thresholdaction**, it is deactivated. In Figure 23-3, the threshold is reactivated when the amount of free space increases by *@@thresh_hysteresis* pages:

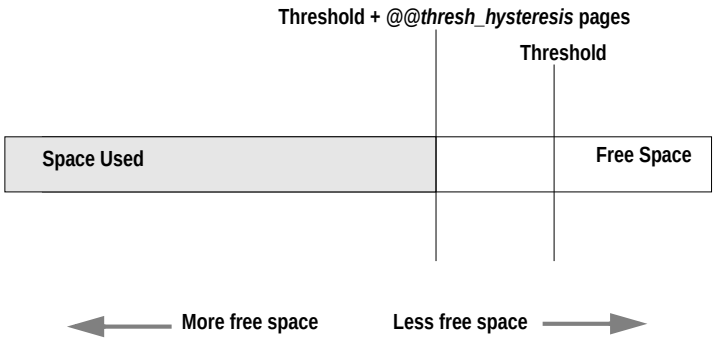


Figure 23-3: Free space must rise by @@thresh_hysteresis to reactivate threshold

Aborting or Suspending Processes

By design, the last-chance threshold allows enough free log space to record a **dump transaction** command. There may not be enough room to record additional user transactions against the database.

When the last-chance threshold is crossed, Adaptive Server suspends user processes and displays the message:

```
Space available in the log segment has fallen
critically low in database 'mydb'. All future
modifications to this database will be suspended
until the log is successfully dumped and space
becomes available.
```

Only commands that are not recorded in the transaction log (**select**, **readtext**, and **writetext)** and commands that might be necessary to free additional log space (**dump transaction**, **dump database**, and **alter database**) can be executed.

Aborting Processes

To abort user processes rather than suspend them, use the **abort tran on log full** option of **sp_dboption** followed by the **checkpoint** command. For example, to abort processes in *mydb* when the last-chance threshold is crossed:

```
sp_dboption mydb, "abort tran on log full", true
use mydb
checkpoint
```

Waking Suspended Processes

Once the **dump transaction** command frees sufficient log space, suspended processes automatically awaken and complete. If **writetext** or **select into** has resulted in unlogged changes to the database since the last backup, the last-chance threshold procedure cannot execute a **dump transaction** command. When this occurs, make a copy of the database with **dump database**, then truncate the log with **dump transaction**.

If this does not free enough space to awaken the suspended processes, it may be necessary to increase the size of the transaction log. Use the **log on** option of the **alter database** command to allocate additional log space.

As a last resort, System Administrators can use the **sp_who** command to determine which processes are in a log suspend status and then use the **kill** command to kill the sleeping process.

Adding, Changing, and Deleting Thresholds

The Database Owner or System Administrator can create additional thresholds to monitor free space on any segment in the database. Additional thresholds are called **free-space thresholds**. Each database can have up to 256 thresholds, including the last-chance threshold.

The **sp_addthreshold**, **sp_modifythreshold**, and **sp_dropthreshold** system procedures allow you to create, change, and delete thresholds. To prevent users from accidentally affecting thresholds in the wrong database, these procedures require that you specify the name of the current database.

Displaying Information About Existing Thresholds

Use the system procedure **sp_helpthreshold** for information about all thresholds in a database. Use **sp_helpthreshold** *segment_name* for information about the thresholds on a particular segment.

The following example displays information about the thresholds on the database's default segment. Since "default" is a reserved word, it must be enclosed in quotation marks. The output of **sp_helpthreshold** shows that there is one threshold on this segment set at 200 pages. The 0 in the "last chance" column indicates that this is not a last-chance threshold but is a free-space threshold:

```
sp_helpthreshold "default"
segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   ------------------
default        200                     0   space_dataseg

(1 row affected, return status = 0)
```

Thresholds and System Tables

The system table *systhresholds* holds information about thresholds. The system procedure **sp_helpthreshold** uses this table to provide its information. In addition to information about segment name, free page, last-chance status, and the name of the threshold procedure,

the table also records the server user ID of the user who created the threshold and the roles of the user who created the threshold had at the moment the threshold was created.

Adaptive Server gets information about how much free space is remaining in a segment—and whether to activate a threshold—from the built-in system function **curunreservedpgs().**

Adding a Free-Space Threshold

Use the system procedure **sp_addthreshold** to create free-space thresholds. Its syntax is:

```
sp_addthreshold dbname, segname, free_space, proc_name
```

The *dbname* must specify the name of the current database. The remaining parameters specify the segment whose free space is being monitored, the size of the threshold in database pages, and the name of a stored procedure.

When the amount of free space on the segment falls below the threshold, an internal Adaptive Server process executes the associated procedure. This process has the permissions of the user who created the threshold when he or she executed **sp_addthreshold**, less any permissions that have since been revoked.

Thresholds can execute a procedure in the same database, in another user database, in *sybsystemprocs* or in *master*. They can also call a remote procedure on an Open Server. **sp_addthreshold** does not verify that the threshold procedure exists when you create the threshold.

Changing a Free-Space Threshold

Use the system procedure **sp_modifythreshold** to associate a free-space threshold with a new threshold procedure, free-space value, or segment. **sp_modifythreshold** drops the existing threshold and creates a new one in its place. Its syntax is:

```
sp_modifythreshold dbname, segname, free_space
   [, new_proc_name [, new_free_space
   [, new_segname]]]
```

where *dbname* is the name of the current database, and *segname* and *free_space* identify the threshold that you want to change.

For example, to execute a threshold procedure when free space on the segment falls below 175 pages rather than below 200 pages, enter:

```
sp_modifythreshold mydb, "default", 200, NULL, 175
```

In this example, NULL acts as a placeholder so that *new_free_space* falls in the correct place in the parameter list. The name of the threshold procedure is not changed.

The person who modifies the threshold becomes the new threshold owner. When the amount of free space on the segment falls below the threshold, Adaptive Server executes the threshold procedure with the owner's permissions at the time he or she executed **sp_modifythreshold**, less any permissions that have since been revoked.

Specifying a New Last-Chance Threshold Procedure

You can use **sp_modifythreshold** to change the name of the procedure associated with the last-chance threshold. You **cannot** use it to change the amount of free space or the segment name for the last-chance threshold.

sp_modifythreshold requires that you specify the number of free pages associated with the last-chance threshold. Use **sp_helpthreshold** to determine this value.

The following example displays information about the last-chance threshold, and then specifies a new procedure, **sp_new_thresh_proc**, to execute when the threshold is crossed:

```
sp_helpthreshold logsegment
segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   -------------------
logsegment     40                      1   sp_thresholdaction

(1 row affected, return status = 0)
sp_modifythreshold mydb, logsegment, 40,
sp_new_thresh_proc
```

Dropping a Threshold

Use the system procedure **sp_dropthreshold** to remove a free-space threshold from a segment. Its syntax is:

```
sp_dropthreshold dbame, segname, free_space
```

The *dbname* must specify the name of the current database. You must specify both the segment name and the number of free pages, since there can be several thresholds on a particular segment. For example:

```
sp_dropthreshold mydb, "default", 200
```

Creating a Free-Space Threshold for the Log Segment

When the last-chance threshold is crossed, all transactions are aborted or suspended until sufficient log space is freed. In a production environment, this can have a heavy impact on users. Adding a correctly placed free-space threshold on your log segment can minimize the chances of crossing the last-chance threshold (and blocking user transactions).

The additional threshold should dump the transaction log often enough that the last-chance threshold is rarely crossed. It should not dump it so often that restoring the database requires the loading of too many tapes.

This section helps you determine the best place for a second log threshold. It starts by adding a threshold with a *free_space* value set at 45 percent of log size and adjusts this threshold based on space usage at your site.

Adding a Log Threshold at 45 Percent of Log Size

Use the following procedure to add a log threshold with a *free_space* value set at 45 percent of log size:

1. Use the following query to determine the log size in pages:

   ```
   select sum(size)
   from master..sysusages
   where dbid = db_id("database_name")
   and (segmap & 4) = 4
   ```

2. Use **sp_addthreshold** to add a new threshold with a *free_space* value set at 45 percent. For example, if the log's capacity is 2048 pages, add a threshold with a *free_space* value of 922 pages:

   ```
   sp_addthreshold mydb, logsegment, 922, thresh_proc
   ```

3. Use the **create procedure** statement to create a simple threshold procedure that dumps the transaction log to the appropriate devices. For more information about creating threshold procedures, see "Creating Threshold Procedures" on page 23-12.

Testing and Adjusting the New Threshold

Use the **dump transaction** command to make sure your transaction log is less than 55 percent full. Then use the following procedure to test the new threshold:

1. Fill the transaction log by simulating routine user action. Use automated scripts that perform typical transactions at the projected rate.

   When the 45 percent free-space threshold is crossed, your threshold procedure will dump the transaction log. Since this is not a last-chance threshold, transactions will not be suspended or aborted; the log will continue to grow during the dump.

2. While the dump is in progress, use **sp_helpsegment** to monitor space usage on the log segment. Record the maximum size of the transaction log just before the dump completes.

3. If considerable space was left in the log when the dump completed, you may not need to dump the transaction log so soon, as shown in Figure 23-4:

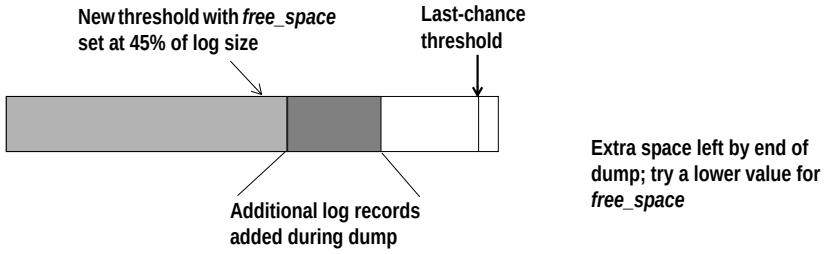


Figure 23-4: Transaction log with additional threshold at 45 percent

Try waiting until only 25 percent of log space remains, as shown in Figure 23-5:

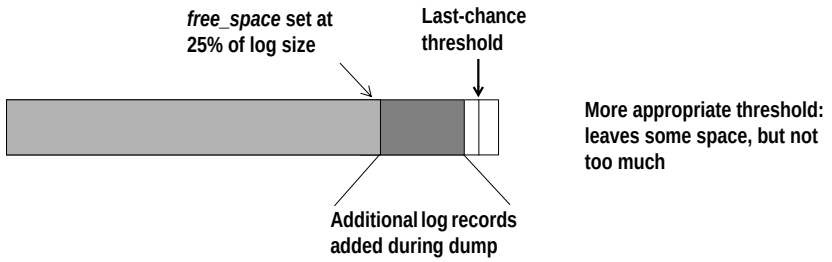


Figure 23-5: Moving threshold leaves less free space after dump

Use **sp_modifythreshold** to adjust the *free_space* value to 25 percent of the log size. For example:

```
sp_modifythreshold mydb, logsegment, 512,
    thresh_proc
```

4. Dump the transaction log and test the new *free_space* value. If the last-chance threshold is crossed before the dump completes, you are not beginning the **dump transaction** soon enough, as shown in Figure 23-6:

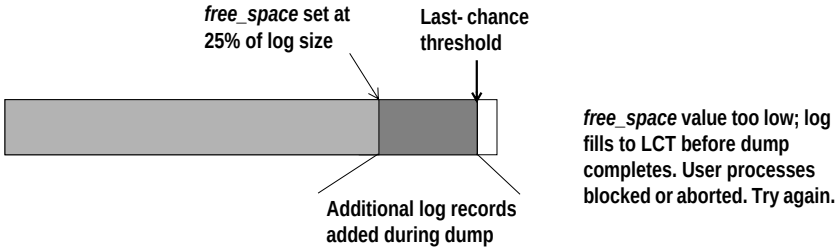


Figure 23-6: Additional log threshold does not begin dump early enough

25 percent free space is not enough. Try initiating the dump transaction when the log has 37.5 percent free space, as shown in Figure 23-7:

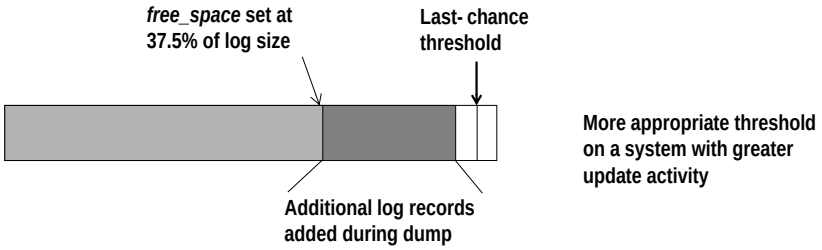


Figure 23-7: Moving threshold leaves enough free space to complete dump

Use **sp_modifythreshold** to change the *free_space* value to 37.5 percent of log capacity. For example:

```
sp_modifythreshold mydb, logsegment, 768,
    thresh_proc
```

Creating Additional Thresholds on Other Segments

You can create free-space thresholds on data segments as well as on log segments. For example, you might create a free-space threshold on the default segment used to store tables and indexes. You would also create an associated stored procedure to print messages in your error log when space on the *default* segment falls below this threshold. If you monitor the error log for these messages, you can add space to the database device when you need it—before your users encounter problems.

The following example creates a free-space threshold on the *default* segment of *mydb*. When the free space on this segment falls below 200 pages, Adaptive Server executes the procedure *space_dataseg*:

```
sp_addthreshold mydb, "default", 200, space_dataseg
```

Determining Threshold Placement

Each new threshold must be at least 2 times *@@thresh_hysteresis* pages from the next closest threshold, as shown in Figure 23-8:

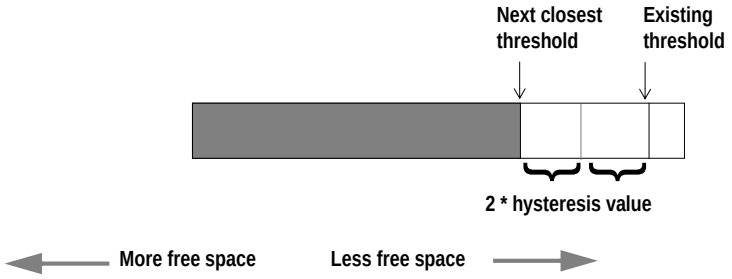


Figure 23-8: Determining where to place a threshold

Use this command:

```
select @@thresh_hysteresis
```

to see the hysteresis value for a database.

In the following example, a segment has a threshold set at 100 pages. The hysteresis value for the database is 64 pages. The next threshold must be at least 100 + (2 * 64), or 228 pages.

```
select @@thresh_hysteresis

-----------
         64

sp_addthreshold user_log_dev, 228,
    sp_thresholdaction
```

Creating Threshold Procedures

Sybase does not supply **sp_thresholdaction** or other threshold procedures. You must create these procedures yourself to ensure that they are tailored to your site's needs.

Suggested actions for **sp_thresholdaction** include writing to the server's error log and dumping the transaction log to increase the amount of log space. You can also design the threshold procedure to execute remote procedure calls to an Open Server, sending mail to the appropriate individuals when a threshold is crossed.

This section provides some guidelines for writing threshold procedures. It includes two sample procedures.

Declaring Procedure Parameters

Adaptive Server passes four parameters to a threshold procedure:

- *@dbname, varchar(30),* which contains the database name.
- *@segmentname, varchar(30),* which contains the segment name.
- *@space_left, int,* which contains the space-left value for the threshold.
- *@status, int,* which has a value of 1 for last-chance thresholds and 0 for other thresholds.

These parameters are passed by position rather than by name. Your procedure can use other names for these parameters, but must declare them in the order shown and with the datatypes shown.

Generating Error Log Messages

Executing a threshold procedure does not automatically generate log messages. If your procedure does not contain a **print** or **raiserror** statement, the error log will not contain any record of the threshold event. You should include a **print** statement near the beginning of your procedure to record the database name, segment name, and threshold size in the error log.

The process that executes threshold procedures is an internal Adaptive Server process. It does not have an associated user terminal or network connection. If you test your threshold procedures by executing them directly (that is, using **execute *procedure_name***) during a terminal session, you see the output from the **print** and **raiserror** messages on your screen. When the same procedures are executed by reaching a threshold, the messages go to the error log. The messages in the log include the date and time.

For example, if **sp_thresholdaction** includes this statement:

```
print "LOG DUMP: log for '%1!' dumped", @dbname
```

Adaptive Server writes this message to the error log:

```
00: 92/09/04 15:44:23.04 server: background task message: LOG
DUMP: log for 'pubs2' dumped
```

Dumping the Transaction Log

If your **sp_thresholdaction** procedure includes a **dump transaction** command, Adaptive Server dumps the log to the devices named in

the procedure. The **dump transaction** command truncates the transaction log by removing all pages from the beginning of the log, up to the page just before the page that contains an uncommitted transaction record.

Once there is enough log space, suspended transactions are awakened. If you abort transactions rather than suspending them, users must resubmit them.

Generally, dumping to a disk is **not** recommended, especially to a disk that is on the same machine or the same disk controller as the database disk. However, since threshold-initiated dumps can take place at any time, you may want to dump to disk and then copy the resulting files to offline media. (You will have to copy the files back to the disk in order to reload them.)

Your choice will depend on:

- Whether you have a dedicated dump device online, loaded and ready to receive dumped data
- Whether you have operators available to mount tape volumes during the times when your database is available
- The size of your transaction log
- Your transaction rate
- Your regular schedule for dumping databases and transaction logs
- Available disk space
- Other site-specific dump resources and constraints

A Simple Threshold Procedure

Following is a simple procedure that dumps the transaction log and prints a message to the error log. Because this procedure uses a variable (*@dbname*) for the database name, it can be used for all databases in Adaptive Server:

```
create procedure sp_thresholdaction
    @dbname varchar(30),
    @segmentname varchar(30),
    @free_space int,
    @status int
as
dump transaction @dbname
    to tapedump1
print "LOG DUMP: '%1!' for '%2!' dumped",
        @segmentname, @dbname
```

A More Complex Procedure

The following threshold procedure performs different actions, depending on the value of the parameters passed to it. Its conditional logic allows it to be used with both log and data segments.

The procedure:

- Prints a "LOG FULL" message if the procedure was called as the result of reaching the log's last-chance threshold. The status bit is 1 for the last-chance threshold and 0 for all other thresholds. The test **if (@status&1) = 1** returns a value of "true" only for the last-chance threshold.
- Verifies that the segment name provided is the log segment. The segment ID for the log segment is always 2, even if the name has been changed.
- Prints "before" and "after" size information on the transaction log. If the log did not shrink significantly, a long-running transaction may be causing the log to fill.
- Prints the time the transaction log dump started and stopped, helping gather data about dump durations.
- Prints a message in the error log if the threshold is not on the log segment. The message gives the database name, the segment name and the threshold size, letting you know that the data segment of a database is filling up.

```
create procedure sp_thresholdaction
     @dbname           varchar(30),
     @segmentname      varchar(30),
     @space_left       int,
     @status           int
as
declare @devname varchar(100),
           @before_size int,
           @after_size int,
           @before_time datetime,
           @after_time datetime,
           @error int

/*
** if this is a last-chance threshold, print a LOG FULL msg
** @status is 1 for last-chance thresholds,0 for all others
*/
if (@status&1) = 1
begin
      print "LOG FULL: database '%1!'", @dbname
end

/*
** if the segment is the logsegment, dump the log
** log segment is always "2" in syssegments
*/
if @segmentname = (select name from syssegments
               where segment = 2)
begin
     /* get the time and log size
     ** just before the dump starts
     */
     select  @before_time = getdate(),
        @before_size = reserved_pgs(id, doampg)
      from sysindexes
      where sysindexes.name = "syslogs"

      print "LOG DUMP: database '%1!', threshold '%2!'",
        @dbname, @space_left

      select @devname = "/backup/" + @dbname + "_" +
        convert(char(8), getdate(),4) + "_" +
        convert(char(8), getdate(), 8)

      dump transaction @dbname to @devname
```

```
        /* error checking */
        select @error = @@error
        if @error != 0
        begin
             print "LOG DUMP ERROR: %1!", @error
        end

        /* get size of log and time after dump */
        select @after_time = getdate(),
             @after_size = reserved_pgs(id, doampg)
             from sysindexes
             where sysindexes.name = "syslogs"

        /* print messages to error log */
        print "LOG DUMPED TO: device '%1!", @devname
        print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
             @before_size, @after_size
        print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time
end          /* end of 'if segment = 2' section */
else         /* this is a data segment, print a message */
begin
        print "THRESHOLD WARNING: database '%1!', segment '%2!' at
'%3!' pages", @dbname, @segmentname, @space_left
end
```

Deciding Where to Put a Threshold Procedure

Although you can create a separate procedure to dump the transaction log for each threshold, it is easier to create a single threshold procedure that is executed by all log segment thresholds. When the amount of free space on a segment falls below a threshold, Adaptive Server reads the *systhresholds* table in the affected database for the name of the associated stored procedure. The entry in *systhresholds* can specify any of the following:

- A remote procedure call to an Open Server
- A procedure name qualified by a database name (for example, *sybsystemprocs.dbo.sp_thresholdaction*)
- An unqualified procedure name

If the procedure name does not include a database qualifier, Adaptive Server looks in the database where the shortage of space occurred. If it cannot find the procedure there, and if the procedure name begins with the characters "sp_", Adaptive Server looks for the procedure in the *sybsystemprocs* database and then in *master* database.

If Adaptive Server cannot find the threshold procedure, or cannot execute it, it prints a message in the error log.

Disabling Free-Space Accounting for Data Segments

Use the **no free space acctg** option to **sp_dboption**, followed by the **checkpoint** command, to disable free-space accounting on non-log segments. You **cannot** disable free-space accounting on the log segment.

When you disable free-space accounting, only the thresholds on your log segment monitor space usage. Crossing thresholds on your data segments will not cause your threshold procedures to be executed. Disabling free-space accounting speeds recovery time because free-space counts are not recomputed during recovery for any segment except the log segment.

The following example turns off free-space accounting for the *production* database:

```
sp_dboption production,
     "no free space acctg", true
```

◆ ***WARNING!***

System procedures cannot provide accurate information about space allocation when free-space accounting is disabled.

Index

The index is divided into three sections:

- Symbols

  Indexes each of the symbols used in this manual.
- Numerics

  Indexes entries that begin numerically.
- Subjects

  Indexes subjects alphabetically.

Page numbers in **bold** are primary references.

Symbols

, (comma)
  in SQL statements xxxvi
{} (curly braces)
  in SQL statements xxxvii
... (ellipsis) in SQL statements xxxviii
% (percent sign)
  error message placeholder 4-3
?? (question marks)
  for suspect characters 14-4, 14-6
" " (quotation marks)
  enclosing parameter values 1-7
  enclosing values 17-23
[ ] (square brackets)
  in SQL statements xxxvii

Numerics

7-bit ASCII character data, character set conversion for 13-8, 14-1, 14-5

A

abort tran on log full database option 16-2, 23-4
Access
  ANSI restrictions on tapes 21-22
  remote 20-30
Adding
  database devices 6-1 to 6-6, 11-139
  date strings 13-15 to 13-16
  dump devices 20-33
  months of the year 13-15 to 13-16
  named time ranges 12-4 to 12-5
  resource limits 12-17 to 12-19
  space to a database 15-12 to 15-14
  thresholds 23-5 to 23-11
  users to a database 11-139
additional network memory configuration parameter 11-89
Address, Server 1-11
address lock spinlock ratio configuration parameter 11-51
Affinity
  process 10-2
  process to engine 10-4
Aliases
  device names 20-32
Aliases, user
  database ownership transfer and 15-12

Allocation errors, correcting with **dbcc** 18-16
Allocation for *dbccdb* database 18-34
Allocation pages 6-2, 15-15, 18-2
  dbcc tablealloc and 18-17
Allocation units 6-2, 15-15, 18-2
  See also Size; Space allocation
  on master device 22-7
  recovery and 21-41
allow backward scans configuration parameter 11-96
allow nested triggers configuration parameter 11-97
allow nulls by default database option 16-3
allow procedure grouping configuration parameter 11-129
allow remote access configuration parameter 11-72
  Backup Server and 20-30
allow resource limits configuration parameter 11-97
allow sql server async i/o configuration parameter 11-36
allow updates configuration parameter (now called **allow updates to system tables**) 1-9, 11-98
allow updates to system tables configuration parameter 1-9, 11-98
alter database command 15-12 to 15-14
  See also **create database** command
  backing up *master* after 20-36
  for load option 15-14
  omitting database device and 6-7, 6-8
  size of database and 15-5
  system tables and 5-8, 5-8 to 5-9, 17-19
  with override option 15-14
Alternate languages. *See* Languages, alternate
ANSI tape label 21-35
Application design 11-139
Applications
  applying resource limits to 12-7
  changing resource limits on 12-21
  dropping resource limits from 12-23
  getting resource limit information about 12-19
  identifying usage-heavy 12-8 to 12-10
  memory for 8-1
  modifying resource limits for 12-21
  names of 12-8
Architecture
  Server SMP 10-2 to 10-4
ASCII characters
  character set conversion and 14-2, 14-5
Asynchronous I/O
  device mirroring and 7-7
  enabling 11-36
  limiting Server requests for 11-81
Asynchronous prefetch
  configuring 11-26
  configuring limits 9-23
@@char_convert global variable 13-3
@@client_csid global variable 13-4
@@client_csname global variable 13-4
@@langid global variable 13-6
@@language global variable 13-6
@@max_connections global variable 11-138
@@maxcharlen global variable 13-4
@@ncharsize global variable 13-4
@@rowcount global variable
  resource limits and 12-10
  row count limits and 12-16
@@thresh_hysteresis global variable 23-3
  threshold placement and 23-11
at option 21-10
  dump striping and 21-20
Auditing
  queue, size of 11-130
  sybsecurity database 2-7
auditing configuration parameter 11-129
audit queue size configuration parameter 11-130
Audit trail 2-7
  backtrace of error messages 4-4
auto identity database option 16-3
Automatic operations

checkpoints 20-4
primary and secondary database dumps 16-5
recovery 20-6 to 20-7

B

Backtracing errors. *See* Error logs
Backup commands. *See* **dump database; dump transaction**
Backup devices. *See* Dump devices
Backups 3-7 to 3-10, 20-1 to 20-39
changes to user IDs 20-36
hints 3-7 to 3-10
multiple databases to a single volume 21-23
planning 20-39
preventing tape overwrites 20-31
remote 20-25
Backup Server 20-25 to 20-28
checking with **showserver** 22-13
device names and 20-32
dump striping and 21-18
error messages 4-13
interfaces file and 20-28
location of 20-28
messages 21-24 to 21-25
multi-file media and tape expiration 21-22
network name 22-12
reading copied dumps 20-31
remote 21-11
remote access 20-30
requirements for dumps 20-28
shutting down 4-24
starting 20-30
sysservers table 20-28
tape retention in days configuration parameter 11-25
volume handling messages 21-34 to 21-39
Batch processing
active time ranges and 12-7
limiting elapsed time 12-15
resource limit scope and 12-12
bcp (bulk copy utility)
character set conversion and 14-5, 14-7
dump database command and 20-35
fast version 16-6
select into/bulkcopy/pllsort and 16-6
sort order changes and 13-9
Binary expressions xxxix
Binary sort order of character sets
character set changes and database dumps 13-9
dbcc checktable and 18-12
Blocking process
server engine task 10-4 to 10-5
Block size
database device 6-4
database dumps and loads 21-13
dump device 20-26
blocksize option 21-11 to 21-13
Brackets. *See* Square brackets [ ]
buildmaster utility command 10-6, 22-5
See also Utility Programs manual
Bytes
block size 21-13
character 14-2
procedure (proc) buffers 8-11
tape capacity 21-13

C

Cache, procedure 11-29
Caches, data
database integrity errors and 4-12
loading databases and 21-49 to 21-51
Caches, data. *See* Data caches
Caches, metadata. *See* Metadata caches
calignment configuration parameter (now called **memory alignment boundary**) 11-26
capacity option 21-11 to 21-13
Cartesian product 12-1
Case sensitivity
in SQL xxxvii

cclkrate configuration parameter (now called **sql server clock tick length**) 11-126
cfgcprot configuration parameter (now called **permission cache entries**) 11-139
cguardsz configuration parameter (now called **stack guard size**) 11-140
Chains of pages
text or *image* data 17-15
Changing
See also Updating
configuration parameters 11-18
database options 16-1 to 16-9
Database Owners 15-12
database size 15-12 to 15-14
hardware 7-8
master database 20-36
named time ranges 12-6
resource limits 12-21 to 12-22
sort order 13-12, 18-13
space allocation once assigned 15-7, 15-12
system tables, dangers of 1-7, 1-9, 22-7
thresholds 23-6
@@char_convert global variable 13-3
char_convert option, **set** 14-4, 14-6, 14-6 to 14-7
Character expressions xxxix
Characters
that cannot be converted 14-2 to 14-3
Character sets 11-48
See also Japanese character sets
changing 13-6 to 13-8
conversion between client and file system 14-7
conversion between client and server 13-6, 14-1 to 14-7
conversion between client and terminal 14-7
conversion errors 13-7, 14-2 to 14-4, 14-6
conversion paths supported 14-1 to 14-3
database dumps and 21-26
definition files 13-2
disabling conversion 13-6
errors when changing 13-7
getting information about 13-8
ID number 11-48
iso_1 14-1
multibyte 13-13
multibyte, changing to 13-11, 13-14
null 14-5
passwords and 14-5
reindexing after configuring 13-10 to 13-14
set char_convert 14-4 to 14-7
setting up for conversion 14-4 to 14-7
Sybase Character Sets product 13-1
translation files, terminal-specific 13-2, 14-8
upgrading text values after changing 13-13
charset.loc file 13-2
charsets directory 13-3
checkalloc option, **dbcc** 18-4, 18-13, 18-19
checkcatalog option, **dbcc** 17-16, 18-18, 18-20
checkdb option, **dbcc** 18-13, 18-19
checkpoint command 20-6
setting database options and 16-8
Checkpoint process 11-23, 20-3 to 20-6
clearing transaction logs 20-4
no chkpt on recovery database option 16-5
recovery interval parameter and 11-24
transaction log and 20-6
trunc log on chkpt database option 11-23, 16-7, 20-4
checkstorage option, **dbcc** 18-7, 18-19
checktable option, **dbcc** 13-12, 18-11, 18-19
transaction log size and 15-9
cindextrips configuration parameter (now called **number of index trips**) 11-27

cis bulk insert batch size configuration parameter 11-31
cis connect timeout configuration parameter 11-32
cis cursor rows configuration parameter 11-32
cis packet size configuration parameter 11-33
cis rpc handling configuration parameter 11-34
Client
character set conversion 13-6, 14-7
@@client_csid global variable 13-4
@@client_csname global variable 13-4
Closed Problem Reports 4-25
Clustered indexes
migration of tables to 17-15, 17-24
segments and 17-15, 17-24
cmaxnetworks configuration parameter (now called **max number network listeners**) 11-77
cmaxscheds configuration parameter (now called **i/o polling process count**) 11-109
cnalarm configuration parameter (now called **number of alarms**) 11-113
cnblkio configuration parameter (now called **disk i/o structures**) 11-37
cnmaxaio_engine configuration parameter (now called **max async i/os per engine**) 11-81
cnmaxaio_server configuration parameter (now called **max async i/os per server**) 11-81
cnmbox configuration parameter (now called **number of mailboxes**) 11-117
cnmsg configuration parameter (now called **number of messages**) 11-118
cntrltype option
disk init 6-5
coamtrips configuration parameter (now called **number of oam trips**) 11-28
Column name
unqualified 4-9
Comma (,)
in SQL statements xxxvi
common.loc file 13-4
Comparing values
datatype problems 4-8
Compiled objects
procedure (proc) buffers for 8-11
Configuration (Server)
See also Configuration parameters
character sets 13-6 to 13-8
configuration file and caches 9-31
message language 13-6 to 13-10
named data caches 9-31
resource limits 12-2
SMP environment 10-5 to 10-9
sort orders 13-6 to 13-13
Configuration file
default name and location 11-6
specifying at start-up 11-10
storage of configured value 11-6
configuration file configuration parameter 11-47
Configuration parameters 11-21 to 11-146
default settings of 11-5
help information on 8-6, 11-7
listing values of 11-8
remote logins and 11-72
resource limits 12-2
Connecting to Adaptive Server 1-11
Connections
interfaces files 1-11
maximum user number 11-138
Consistency
checking databases 3-10, 20-17
Constants xxxviii
contiguous option (OpenVMS)
disk init 6-5
Conventions
Transact-SQL syntax xxxvi to xxxix
used in manuals xxxvi
Copying
dump files and disks 20-31

Copying selected data. *See* **insert** command; **select** command
Corrupt databases
  assessing number of suspect pages 20-16
  isolating suspect pages 20-8
  recovery fault isolation mode 20-7
  system compared to user 20-8
Corrupt pages
  assessing 20-16
  isolating on recovery 20-7 to 20-17
  listing 20-10
Cost
  I/O 12-14
cp437 character set 11-48
cp850 character set 11-48
cpreallocext configuration parameter (now called **number of pre-allocated extents**) 11-118
CPR files 4-25
cpu accounting flush interval configuration parameter 11-99
cpu flush configuration parameter (now called **cpu accounting flush interval**) 11-99
cpu grace time configuration parameter 11-100
CPU usage
  monitoring 10-7
  number of engines and 10-6 to 10-7
  symmetric processing and 10-2 to 10-4
create database command 15-3 to 15-12
  allocating storage space with 15-5
  backing up *master* after 20-36
  default database size configuration parameter and 11-101, 15-6
  for load option and 15-10 to 15-11, 15-14
  log on option 5-4, 15-5, 15-7
  model database and 2-4
  omitting database device and 6-7, 6-8
  omitting **log on** option 15-9
  omitting **on** 15-6
  on keyword 15-5
  permission 15-2
  size parameter 15-6
  system tables and 1-5, 17-19
  with override option 15-11, 15-14
create index command 5-3, 5-9, 17-10
  database dumping and 20-34
  moving tables with 17-25
create procedure command 1-9
create table command 5-3, 17-10
  clustered indexes and 17-25
Creating
  database objects 5-3
  database objects on segments 17-10
  databases 15-3, 15-12
  logical names 20-32
  master database 5-5
  model database 5-5
  named time ranges 12-4 to 12-5
  resource limits 12-17 to 12-19
  segments 5-5, 17-7
  stored procedures 1-9
  system procedures 1-9
  system tables 1-5
  tempdb database 5-5
  thresholds 23-5 to 23-11
  user-defined error messages 4-6
Cross-database referential integrity constraints
  loading databases and 21-52
cs_connection command, **number of user connections** and 11-139
cschedspins configuration parameter (now called **runnable process search count**) 11-123
csortbufsize configuration parameter (now called **number of sort buffers**) 11-119
ctimemax configuration parameter (now called **cpu grace time**) 11-100
Curly braces ({})
  in SQL statements xxxvii
current audit table configuration parameter 11-131

Current database 4-8
Current log. *See* Transaction logs
Cursors
  limiting the I/O cost of 12-14
  limiting the number of rows returned 12-17
  row count, setting 11-32

D

Damage symptoms, *master* database. *See* *master* database
Data
  losing unlogged 16-6
Database administration 1-1 to 1-2
Database corruption
  caused by copying database devices 20-17
Database devices 6-1
  See also Disk mirroring; Dump devices; Master device
  adding 6-1 to 6-6
  assigning databases to 15-5, 15-14, 21-43
  default 2-2, 6-7 to 6-9, 15-7
  dropping 6-7
  fragments 5-8
  information about 6-6, 15-18
  initializing 6-1 to 6-6
  names of 5-5, 6-3
  numbering 6-3, 15-4
  number of Server-usable 8-13, 11-38
  performance tuning 17-4 to 17-25
  placing objects on 5-5, 17-10
  recovery and 22-21
  transaction logs on separate 7-2
  unmirroring and 7-8
Database device space. *See* Segments; Space allocation
Database objects
  assigning to devices 5-4, 17-3, 17-10
  controlling user creation of 2-3, 20-36
  creating 2-3, 5-3
  dropping 15-15
  dropping segments and 17-16
  errors affecting 4-11
  finding 4-8
  maximum number of open 11-66
  performance tuning and 17-4
  placement on segments 17-3, 17-10, 17-12, 21-44
  space used by 15-20
Database options 16-1 to 16-10
  changing 16-8 to 16-9
  listing 16-2
  setting 16-2 to 16-7
  showing settings 16-2
Database Owners
  changing 15-12
  error responsibilities of 4-7, 4-10
  login name 1-2
Databases
  adding users 15-5
  assigning to database devices 15-5
  backing up 2-5, 3-8, 18-22
  backup/log interactions and 20-6
  binding to data caches 9-15
  checkalloc option (**dbcc**) 18-13
  checkdb option (**dbcc**) 18-13, 18-19
  checkstorage option (**dbcc**) 18-7
  checktable option (**dbcc**) 18-11
  creating user 15-3 to 15-12
  creating with separate log segment 15-7
  default 2-4
  default size 15-6
  default storage for 2-2, 6-7
  dropping 15-14, 18-26, 21-42
  dropping users from 15-5
  dumping 3-8, 18-22, 20-17
  errors affecting 4-11
  increasing size of 15-12
  indexalloc option (**dbcc**) 18-15
  information on storage space used 15-19
  integrity concerns 4-12, 18-2 to 18-22
  loading 21-44

loading after character set change 13-9
loading after sort order change 13-9
maintenance of 18-1 to 18-22
monitoring space used by 15-20
moving to different machines 15-10 to 15-11, 20-20, 21-40
name 15-3
new 2-4
number of open 11-62
options 16-1 to 16-9
recovering 21-39
removing and repairing damaged 21-42
running out of space in 21-31
sequence numbers for recovery 16-5
size 2-4
storage information 15-15
system 2-1 to 2-9
tablealloc option (**dbcc**) 18-16
upgrading database dumps 21-46
user 15-2
Database segments. *See* Segments
database size configuration parameter (now called **default database size**) 11-101
Data caches
changing bindings 9-16
changing size 9-24
changing type 9-10
command summary 9-3 to 9-4
configuration file and 9-31
configuring 9-3, 9-31 to 9-36
database integrity errors and 4-12
dbcc and 18-19
default 9-1, 9-8, 9-35
dropping 9-26
dropping bindings 9-19
I/O size and 9-35
information about 9-4 to 9-6, 9-17
overhead 9-17
sizing 9-35
status 9-7
Data dictionary. *See* System tables
Data rows
checking with **dbcc** commands 18-11
Date parts
alternate language 13-15
Dates
adding date parts 13-15
alternate language 13-15
display formats 13-4
format in error messages 4-5
Days
alternate language 13-15
dbcc (Database Consistency Checker) 3-10, 18-1 to 18-45
backups and 20-17
checks performed by 18-6
commands compared 18-19
database damage and 4-7, 4-11, 18-2
database maintenance and 18-1 to 18-22, 21-39
output of 18-23 to 18-24
rebuilding indexes with 13-12
reports 18-23
scheduling 18-20 to 18-22
when to use 4-11
dbccdb database
consistency checks for 18-40
creating workspaces in 18-30
deleting **dbcc checkstorage** history from 18-40
installing 18-35
reporting configuration information from 18-42
reporting fault information from 18-43
dbid column, *sysusages* table 15-16
DB-Library programs
client character set and 14-4
number of user connections and 11-139
dbo use only database option 16-3
"dbo" user name 1-2
dbprocess command, **number of user connections** and 11-139
dbrepair option, **dbcc** 18-26, 21-42
drop database and 18-26

ddl in tran database option 16-3
deadlock checking period configuration parameter 11-52
deadlock retries configuration parameter 11-53
Deadlocks 4-8
  descending scans and 11-96
deckanji character set 11-48
default character set id configuration parameter 11-48
Default database devices 15-7
  designating 6-7
  sample database and 2-2
default database size configuration parameter 11-101, 15-6
default fill factor percent configuration parameter 11-102
default language configuration parameter (now called **default language id**) 11-48
default language id configuration parameter 11-48
default network packet size configuration parameter 11-72
defaulton | defaultoff option, **sp_diskdefault** 6-8
default segment 5-5, 17-2
  cannot drop from a database 17-16
  extending 17-8
  reducing scope 17-9
Default settings
  changing character set 13-7 to 13-14
  changing sort order 13-8 to 13-13
  character set ID number 11-48
  configuration parameters 11-5
  databases 2-4
  database size 15-6
  language 11-48
  memory allocation 11-92
  permissions 2-5
  sort order 11-49
  system databases at installation 5-5
default sortorder id configuration parameter 11-49
defncopy utility command
  See also Utility Programs manual
  character set conversion and 14-5, 14-7
delete command
  transaction log and 20-3
Deleting
  See also Dropping
  files 6-7
density option 21-11 to 21-13
Descending scans
  deadlocks and 11-96
Development server 3-1
Device failure 20-17
  dumping transaction log after 21-2, 21-28
  master device 7-2
  user databases 7-2
Device fragments 5-8
Devices 6-1
  See also Database devices; Dump devices; Master device
  adding 6-1 to 6-6
  aliasing names of 20-32
  dropping 6-7, 21-42
  information listings on 6-6, 21-41
  initializing 6-1 to 6-6
  listing 20-32
  names for physical 6-3, 20-32 to 20-33
  number of user connections and 11-138, 11-139
  operating system constraints 6-4
  splitting tables across 17-12 to 17-13
  using separate 5-4, 15-7, 17-3
devices configuration parameter (now called **number of devices**) 11-38
Directory structure
  character sets 13-3
  internationalization files 13-3
  localization files 13-5
  **.loc* files 13-5
Direct updates
  to system tables 11-98
Dirty buffer grab, wash size and 9-22

Dirty pages 11-23, 20-3
disable character set conversions configuration parameter 11-49
Disabling mirroring. *See* **disk unmirror** command
Disk allocation pieces 15-18
Disk controllers 6-5, 17-5
Disk devices
  See also Database devices; Dump devices; Space allocation
  adding 20-33
  dumping to 20-31
  mirroring 7-1 to 7-10
  unmirroring 7-8
Disk I/O
  configuration parameters for 11-36
  database loads and 11-21
  memory for 8-13
  mirrored devices and 7-5
disk i/o structures configuration parameter 11-37
disk init command 5-2, 5-7, 5-8, 6-1 to 6-6
  allocation and 18-2
  master database backup after 20-36
  mirror devices and 7-6
disk mirror command 5-3, 7-1, 7-6 to 7-10
Disk mirroring 7-1 to 7-13
  asynchronous I/O and 7-7, 7-10
  effect on *sysdevices* 7-8, 7-9, 7-11 to 7-13
  initializing 7-6 to 7-7
  recovery and 5-4
  restarting 7-10
  status in *sysdevices* table 6-7
  tutorial 7-11
  unmirroring and 7-8
  waitfor mirrorexit 7-10
disk refit command 22-22 to 22-23
disk reinit command 22-21
  See also **disk init** command
disk remirror command 7-10
  See also Disk mirroring
Disks. *See* Database devices; Devices; Dump devices
disk unmirror command 7-8
  See also Disk mirroring
drop database command 15-14
  damaged databases and 18-26
  for failed devices 21-42
dropdb option, **dbcc dbrepair** 18-26, 21-42
Dropping
  damaged database 18-26
  database devices 6-7, 15-15, 21-42
  databases 15-14, 18-26, 21-42
  dump devices 6-7, 20-33
  master device from default space pool 6-8
  named time ranges 12-6
  resource limits 12-22 to 12-24
  segment from a database 17-16
  thresholds 23-7
Dump, database 3-8, 18-22, 21-1 to 21-25
  block size 21-13
  database name 21-5, 21-6
  dismounting tapes 21-21
  dump devices 21-7
  dump striping 21-17
  expiration date 21-22
  file name 21-15 to 21-17
  initializing/appending 21-22
  message destination 21-24
  multiple to a single volume 21-23
  rewinding tapes after 21-22
  routine 20-18
  sp_volchanged prompts 21-35 to 21-38
  upgrading user database dumps 21-46
  volume labels 21-14
  volume name 21-14
Dump, transaction log 20-18, 21-1 to 21-25
  database name 21-6
  dismounting tapes 21-21
  dump devices 21-5, 21-7
  dumping after a media failure 21-28
  dump striping 21-17
  file name 21-15 to 21-17
  insufficient log space 21-5

maximizing space 21-30
message destination 21-24
rewinding tapes after 21-22
sp_volchanged prompts 21-35 to 21-38
tape capacity 21-13
volume name 21-14
dump database command 21-1 to 21-46
See also Dump, database
dbcc schedule and 18-22
disk init and 6-2
master database and 3-8
model database and 2-5
permissions for execution 20-25
prohibited in offline databases 20-12
when to use 18-22, 20-33 to 20-38
Dump devices
adding 20-33
disks as 20-31
dropping 6-7, 20-33
files as 20-31
information about 6-6
listing 20-32
logical names for 20-32 to 20-33
maximum allowed 21-18
multiple 21-17 to 21-20
permissions for 20-28
redefining 20-33
specifying 21-5, 21-7
sysdevices table and 5-7, 20-32
tape retention in days and **retaindays** meaningful for 21-22
tapes as 20-31
dump on conditions configuration parameter 11-103
Dump striping 21-17 to 21-20
backing up databases to multiple devices 20-25
dump transaction command 15-7, 15-8, 15-10, 21-2 to 21-46
See also Dump, transaction log
in *master* database 20-37
in *model* database 20-37
permissions for execution 20-25
prohibited in offline databases 20-12
threshold procedures and 23-13
trunc log on chkpt and 16-7, 20-5
with no_log option 20-35, 21-31
with no_truncate option 21-28
with truncate_only option 21-30
dumpvolume option 21-14
Dynamic configuration parameters 11-6

E

Editing. *See* Changing; Updating
Ellipsis (...) in SQL statements xxxviii
enable cis configuration parameter 11-34
enable rep agent threads configuration parameter 11-95
End-of-tape marker 21-13
Engines 10-1
functions and scheduling 10-2 to 10-4
identification numbers 4-5
managing 10-6 to 10-7
number of 10-6 to 10-7, 11-94
tasks of server 10-4 to 10-5
Error logs 3-11, 4-11, 8-10 to 8-11
creation and ownership 4-4
format 4-5
location 1-10
monitoring cache sizes with 8-10
purging 4-5
Error messages 4-2 to 4-12
for allocation errors 18-17
altering Server-provided 4-6, 13-5
character conversion 14-3, 14-6
creating user-defined 4-6
for fatal errors 4-10 to 4-12
for memory use 8-10 to 8-11
numbering of 4-2
severity levels of 4-6 to 4-12
tablealloc allocation 18-23
thresholds and 23-13
user-defined 4-6
Errors
See also Error logs; Error messages
allocation 18-14, 18-16
character conversion 14-2 to 14-4

correcting with **dbcc** 18-16
fatal 4-10 to 4-12
input/output 22-2
logging 4-4
multiple 4-1
reporting of 4-12
segmentation 22-2
Server responses to 4-1 to 4-12
state numbers 4-1
types of information logged 1-10
user 4-7, 4-7 to 4-10
esp execution priority configuration parameter 11-43
esp execution stacksize configuration parameter 11-44
esp unload dll configuration parameter 11-44
Estimated cost
resource limits for I/O 12-11, 12-13
EUC JIS. *See* Japanese character sets
eucjis character set 11-48
event buffers per engine configuration parameter 11-103
event log computer name configuration parameter 11-40
event logging configuration parameter 11-41
executable code size + overhead configuration parameter 11-61
Execution
ESPs and XP Server priority 11-43
resource limits and 12-11
Expired passwords 11-136
Expressions
types of xxxviii to xxxix
Extended stored procedures
configuration parameters 11-43 to 11-46
Extended UNIX character set 11-48
Extending segments 17-8
Extents
I/O size and 9-2
sp_spaceused report and 15-20
space allocation and 18-2

F

Failures, media 4-12
copying log device after 20-18, 21-28 to 21-30
diagnosing 21-39
recovery and 20-17
fast option
dbcc indexalloc 18-15, 18-16, 18-17, 18-20
dbcc tablealloc 18-17, 18-20
Fatal errors
backtrace from kernel 4-4, 4-11
error messages for 4-10 to 4-12
severity levels 19 and up 4-10 to 4-12
File descriptors 11-138
File names
database dumps 21-15 to 21-17
transaction log dumps 21-15 to 21-17
file option 21-15
Files
character set translation (*.xlt*) 13-2
Closed Problem Reports (CPRs) 4-25
deleting 6-7
dump to 20-31
error log 1-10, 4-4
interfaces 1-11
interfaces, and Backup Server 21-11
internationalization 13-2
localization 13-4 to 13-5
mirror device 7-7
System Problem Reports (SPRs) 4-25
Fillfactor
default fill factor percent configuration parameter 11-102
fillfactor configuration parameter (now called **default fill factor percent**) 11-102
Finding
database objects 4-8
fix_text option, **dbcc** 13-13 to 13-14
fix option
dbcc 18-16
dbcc checkalloc 18-14
dbcc indexalloc 18-17
dbcc tablealloc 18-23

using single-user mode 18-16
Floating-point data xxxix
for load option
alter database 15-14
create database 15-10 to 15-11
Formats
date, time, and money 13-4
locale, unsupported 13-15 to 13-16
Formulas
user requirements and 11-138
Fragments, device space 5-8, 15-16, 17-20
freelock transfer block size configuration parameter 11-55
Free space, log segment and 23-1 to 23-18
full option
dbcc indexalloc 18-15, 18-16, 18-17, 18-19
dbcc tablealloc 18-17, 18-19

G

Gaiji. *See*Japanese character sets
global asynch prefetch limit configuration parameter 11-26
grant command
database creation 15-2
group by clause
tempdb database and 2-6
Guest users
sample databases and 2-8

H

Halloween problem
avoiding with **unique auto_identity index** database option 16-8
Hankaku Katakana. *See* Japanese character sets
Hardware
errors 4-12
unmirroring 7-8
Header information
"proc headers" 8-11
headeronly option 20-22, 21-25 to 21-28
housekeeper free write percent configuration parameter 11-104
Housekeeper task
configuring 11-104
Hysteresis value, *@@thresh_hysteresis* global variable 23-11

I

I/O
configuring size 9-11 to 9-15
costing 12-14
devices, disk mirroring to 7-7
errors 22-2
evaluating cost 12-9, 12-13 to 12-15
limiting 12-10
limiting a pre-execution time 12-11
statistics information 12-14
i/o accounting flush interval configuration parameter 11-108
i/o flush configuration parameter (now called **i/o accounting flush interval**) 11-108
i/o polling process count configuration parameter 11-109
IBM character set 11-48
identity burning set factor configuration parameter 11-106
IDENTITY columns
automatic 16-3, 16-8
nonunique indexes 16-5
identity grab size configuration parameter 11-107
identity in nonunique index database option 16-5
IDs, time range 12-4
IDs, user
system procedures and 1-9
image datatype
performance effects of 17-6
storage on separate device 17-15
sysindexes table and 17-15, 17-19
indexalloc option, **dbcc** 18-15, 18-19

Index descriptors
  maximum number open 11-64
Indexes
  assigning to specific segments 17-5, 17-22
  binding to data caches 9-15
  character-based 13-11
  character set changes 13-10 to 13-13
  database dumping after creating 20-34
  default fill factor percent percentage for 11-102
  IDENTITY columns in nonunique 16-5
  integrity checks (**dbcc**) 13-12
  Object Allocation Maps of 11-28, 18-4
  rebuilding 13-11, 20-34
  single device placement for 17-5
  sort order changes 13-10 to 13-13, 18-11
  suspect 4-11, 13-10, 13-11
Information (Server)
  backup devices 20-32
  configuration parameters 11-8
  database devices 6-6, 15-18
  database options 16-2
  database size 15-6, 15-20
  database storage 15-15
  data caches 9-4
  dbcc output 18-23
  device names 20-32
  devices 6-6
  dump devices 6-6, 20-32
  error messages 4-2 to 4-12
  Open Client applications 4-7
  problems 4-4
  resource limits 12-19 to 12-21
  segments 15-18 to 15-22, 17-16 to 17-18, 18-18
  space usage 15-18 to 15-22
  thresholds 23-5
Information messages (Server). *See* Error messages; Severity levels
Initializing
  database devices 6-1 to 6-6
  disk mirrors 7-6 to 7-7
init option 21-20 to 21-24
insert command
  transaction log and 20-3
Installation, Server
  interfaces file 1-11
  status after 5-5
Installing
  sample databases 2-8
installmaster script 22-18
  sybsystemprocs recovery with 20-38
installmodel script 22-16
Insufficient permission 4-8
Insufficient resource errors (Level 17) 4-9
Integer data
  in SQL xxxix
Interfaces file 1-11
  Backup Server 20-28, 21-11
Intermediate display level for configuration parameters 11-16
Internal error, non-fatal 4-10
Internal structures
  memory used for 8-9
Internationalization 13-1
  directory structure for character sets 13-3
  files 13-2
International language support. *See* Character sets; Languages
iso_1 character set 11-48, 14-1
Isolation levels
  level 0 reads 16-5
isql utility command
  character set conversion and 14-5, 14-7
  number of user connections and 11-139
  status and informational messages 4-7
  system administration and 1-2

J

Japanese character sets 11-48, 14-2
  See also Languages, alternate
  conversion between 14-2
  EUC JIS 14-2
  Gaiji 14-3
  Hankaku Katakana 14-3
  Shift-JIS 14-2
  sjis (Shift-JIS) 11-48

K

Kanji. *See* Japanese character sets
Kernel
  error messages 4-4, 4-11
  memory used for 8-9
Keys, table
  on system tables 1-6
kill command 4-14 to 4-17
Known problems 4-25

L

Labels
  dump volumes 21-27
Labels, device. *See* Segments
@@langid global variable 13-6
Language defaults 11-48
  changing user's 13-10
  us_english 11-48, 14-5, 14-7
@@language global variable 13-6
language in cache configuration parameter (now called **number of languages in cache**) 11-50
Languages, alternate 13-2
  See also Character sets; *charset.loc* file; Japanese character sets
  cache 11-50
  configuring 13-9
  date formats in unsupported 13-15
  disabling character set conversion 13-6
  installing on server 13-9
  localization files 13-4 to 13-5
  supported languages 13-1
Last-chance thresholds 23-1 to 23-18
  dumping transaction log 23-13
  number of free pages for 23-7
  procedure, creating 23-12 to 23-18
  procedure, specifying new 23-7
  sample procedure for 23-14 to 23-17
Levels, severity. *See* Severity levels, error
Limit types 12-10, 12-13 to 12-17
  elapsed time 12-15
  I/O cost 12-13
  number of rows returned 12-16
Linkage, page 18-6, 18-11
  See also Pages, data
Listing
  database options 16-2
  dump files on a tape 20-22, 21-25 to 21-28
listonly option 20-22, 21-25 to 21-28
Lists
  sp_volchanged messages 21-35 to 21-38
Load, database 21-44
  automatic remapping 21-44
  data caches and 21-49 to 21-51
  device specification 21-7
  loading databases that use cross-database referential constraints 21-51
  name changes 21-7
  number of large i/o buffers configuration parameter 11-21, 20-38
  sp_volchanged prompts 21-38
Load, transaction log
  device specification 21-7
  order of dumps 21-44
  sp_volchanged prompts 21-38
load database command 21-2 to 21-46
  See also Load, database
  for *master* database 22-13
  for *model* database 22-17
  permissions for execution 20-25
  for *sybsystemprocs* database 22-20
load transaction command 21-2 to 21-46

See also Load, transaction log
permissions for execution 20-25
locales.dat file 13-4
locales directory 13-4, 13-5
Localization 13-1
See also Languages, alternate files for 13-4 to 13-5
Locking
cache binding and 9-30
by **dbcc** commands 13-14, 18-19
SMP environment 10-4
lock promotion HWM configuration parameter 11-110, 11-111
lock promotion LWM configuration parameter 11-111
lock promotion PCT configuration parameter 11-112
Lock promotion thresholds
server-wide 11-110
Locks
quantity of 11-58
locks configuration parameter (now called **number of locks**) 11-58
lock shared memory configuration parameter 11-91
log audit logon failure configuration parameter 11-42
log audit logon success configuration parameter 11-42
Log file. *See* Error logs
Logging
login failures 11-42
successful logins 11-42
Windows NT event log in 11-40, 11-41
Log I/O size 9-14
Logical address 15-17
Logical expressions xxxviii
Logical names 20-32
Logins
active time ranges and 12-7
character set conversion and client 14-4
"dbo" user name 1-2
"sa" 22-9, 22-14
"sa" password 22-6
log on option
alter database 15-13
create database 5-4, 5-8, 15-7, 15-9
Logs. *See* Transaction logs
Log segment
cannot entirely drop from a database 17-16
thresholds for 23-8 to 23-11
logsegment log storage 5-5, 17-2
Losing unlogged data 16-6
lstart column, *sysusages* table 15-18

M

Machine names, character set conversion and 14-5
Machine types, moving databases between 15-10 to 15-11
Macintosh character set 11-48, 14-2
Mail session, starting 11-45
Management, space. *See* Space allocation; Storage management
Mapping
device name to physical name 6-1
master database 1-4, 2-2 to 3-8
See also Disk mirroring; System tables
backing up 3-8, 20-35 to 20-37
backing up transaction log 20-37
changing option settings 16-1
commands that change the 20-36
creating 5-5
damage symptoms 21-39
dumping 20-31
extending with **alter database** 15-13
keys for system tables in 1-6
ownership of 15-12
requirement for dumping to single volume 20-31
sysdevices table 6-6
Master device 2-2, 6-3, 6-6
See also Database devices
buildmaster initialization of 10-6
disk mirroring of 7-2, 7-11

removing from default space pool 6-7, 6-8
sp_diskdefault and 6-8
Master-recover mode 22-6
@@max_connections global variable 11-138
max async i/os per engine configuration parameter 11-81
max async i/os per server configuration parameter 11-81
@@maxcharlen global variable 13-4
max cis remote connections configuration parameter 11-35
max cis remote servers configuration parameter 11-36
max engine freelocks configuration parameter 11-56
maximum dump conditions configuration parameter 11-113
maximum network packet size configuration parameter (now called **max network packet size**) 11-74
max network packet size configuration parameter 11-74
max number network listeners configuration parameter 11-77
max online engines configuration parameter 10-6, 11-94
max roles enabled per user configuration parameter 11-132
max scan parallel degree configuration parameter 11-87
max SQL text monitored configuration parameter 11-91
Memory
See also Space allocation
audit records 11-130
configuring 8-1 to 8-11
error log messages 8-10 to 8-11
executable code 8-8
freeing from XP Server 11-44
how Server uses 8-2 to 8-4
major uses of 8-8 to 8-13
maximizing 8-1 to 8-2
number of open databases and 11-62
optimizing for your system 8-1 to 8-2
parallel processing 8-13
referential integrity 8-15
remote procedure calls 8-15
remote servers 8-14
Server needs for 11-29, 11-92
shared 10-2
in SMP sites 10-8
system procedures used for 8-4 to 8-8
use of by Component Integration Services 8-9
user connections 8-12
worker processes 8-14
memory alignment boundary configuration parameter 11-26
memory configuration parameter (now called **total memory**) 11-92
memory per worker process configuration parameter 11-88
Memory pools
changing size 9-26
configuring 9-11 to 9-15
configuring asynchronous prefetch limits 9-23
configuring wash percentage 9-20 to 9-23
dropping 9-29
Messages
Backup Server 21-24 to 21-25
error 1-10, 4-2 to 4-12
fatal error 1-10
language setting for 13-1
sp_volchanged list 21-35
start-up 1-10
system 4-2 to 4-12
user-defined 4-6
Metadata caches
configuration parameters 11-61 to 11-71
described 8-12
finding usage statistics 8-7
Microsoft character set 11-48
Midpoint between thresholds 23-11

Migration
of tables to clustered indexes 17-15, 17-24
Mirror devices 7-1, 7-6, 7-10
Mirroring. *See* Disk mirroring
Miscellaneous user error 4-9
Mistakes, user. *See* Errors; Severity levels, error
model database 2-4
automatic recovery and 20-6
backing up 20-37
backing up transaction log 20-37
changing database options 16-7
changing options in 16-2
creating 5-5
keys for system tables in 1-6
restoring 22-17 to 22-18
size 6-4, 11-101, 15-6
mode option, **disk unmirror** 7-8
Modifying
named time ranges 12-6
resource limits 12-21 to 12-22
Money
local formats 13-4
Monitoring
CPU usage 10-7
data cache size 8-10
spt_monitor table 1-8
SQL text 11-91
Windows NT Performance Monitor 11-125
Month values
alternate language 13-15
Moving
nonclustered indexes 17-12
tables 17-12, 17-15, 17-24
transaction logs 15-9
mrstart configuration parameter (now called **shared memory starting address**) 11-84
Multibyte character sets 13-13
changing to 13-11, 13-14
default character set id configuration parameter 11-48
incompatible 14-2
Multilingual character set 11-48
Multiprocessor servers. *See* SMP (symmetric multiprocessing) systems
Multiuser environments, splitting tables in 17-12

N

Named cache for **dbcc checkstorage** 18-33
Named time ranges 12-3
adding 12-4 to 12-5
"at all times" 12-3
changing active time ranges 12-7
creating 12-4 to 12-5
dropping 12-6
dropping resource limits using 12-23
modifying 12-6
overlapping 12-3
precedence 12-25
using 12-3 to 12-7
Name of device 6-3
dump device 20-32, 21-41
logical name for physical device 20-33
sysdevices listing 5-7
Names
applications 12-7
ASCII character 14-5
column, in commands 4-9
machine 14-5
partial, in option specification 16-9
segment 17-1, 17-7
system extended stored procedures 1-9
system procedures 1-7
user 14-5
Naming
dump files 21-15 to 21-17
@@ncharsize global variable 13-4
nested trigger configuration parameter (now called **allow nested triggers**) 11-96, 11-97

Networks
backups across 21-10 to 21-11
connections 1-11
dumps across 21-7
dump striping and 21-19
interfaces files 1-11
loads across 21-7
restoring across 22-12
software 3-4
no_log option, **dump transaction** 21-31
no_truncate option, **dump transaction** 21-28 to 21-30
no chkpt on recovery database option 16-5
nodismount option 21-20 to 21-21
nofix option, **dbcc** 18-16
no free space acctg database option 16-6, 23-18
Nonclustered indexes
moving between devices 17-12
Non-logged operations 16-6
Nonstop recovery 5-4, 7-3
noserial option, **disk mirror** 7-7
notify option 21-24 to 21-25
nounload option 21-20 to 21-22
Null passwords 22-6
Number (quantity of)
database devices 8-13, 11-38
dump devices 21-18
engines 10-6, 11-94
errors corrected with **dbcc** 18-17
extents 18-2
locks 11-58
open databases on Server 11-62
open objects 11-66
rows returned 12-10, 12-16
segments 17-2
SMP system engines 10-6
user connections (*@@max_connections*) 11-138
number of alarms configuration parameter 11-113
number of aux scan descriptors configuration parameter 11-114
number of devices configuration parameter 8-13, 11-38
number of index trips configuration parameter 11-27
number of languages in cache configuration parameter 11-50
number of large i/o buffers configuration parameter 11-21, 20-38
number of locks configuration parameter 11-58
number of mailboxes configuration parameter 11-117
number of messages configuration parameter 11-118
number of oam trips configuration parameter 11-28
number of open databases configuration parameter 11-62
number of open indexes configuration parameter 11-64
number of open objects configuration parameter 11-66
number of pre-allocated extents configuration parameter 11-118
number of remote connections configuration parameter 11-78
number of remote logins configuration parameter 11-78
number of remote sites configuration parameter 11-79
number of sort buffers configuration parameter 11-119
number of user connections configuration parameter 11-18, 11-137 to 11-139
Numbers
device 6-3, 15-18
engine 4-5
error message 4-2
segment value 15-17, 21-41
sort order 11-49
status bit (*sysdevices*) 6-7
virtual device 15-18
Numeric expressions xxxix

O

o/s file descriptors configuration parameter 11-83
Object Allocation Map (OAM) pages 18-4
  checking with **dbcc** commands 18-13, 18-16, 18-17
Offline pages 20-8
  effects of 20-12
  listing 20-10
on keyword
  alter database 15-13
  create database 15-5, 15-6
  create index 17-10
  create table 17-10
online database command 20-20, 21-45
  bringing databases online 21-45
  replicated databases 21-45
  restoring a database 20-24, 20-25
  status bits 21-48
  upgrading a database 20-21, 21-47
Open Client applications
  messages 4-7
open databases configuration parameter (now called **number of open databases**) 11-62
open index hash spinlock ratio configuration parameter 11-69
open index spinlock ratio configuration parameter 10-9, 11-70
open objects configuration parameter (now called **number of open objects**) 11-66
open object spinlock ratio configuration parameter 11-71
OpenVMS systems
  contiguous option on 6-5
  foreign device 6-3
  Operator permissions 20-25
  preventing tape dismounts 21-21
  REPLY command 21-34
  volume change messages 21-24
Operating system commands
  executing 1-9
Operating systems
  constraints 6-4
  copy commands corrupting databases 20-17
  failures and automatic recovery 20-6
  file mirroring 7-7
  Sybase task scheduling and 10-2 to 10-4
Operator role
  tasks of 20-25
optimized report
  dbcc indexalloc 18-15, 18-16, 18-17, 18-19
  dbcc tablealloc 18-17, 18-19
Optimizer 9-2
Options
  database 16-1 to 16-10
  unique string for 16-9
order by clause
  tempdb database and 2-6
Order of commands
  clustered index creation and 17-12, 17-13
  for database and log dumps 16-6
  object-level **dbcc** checking 18-22
Overflow errors
  Server stack 11-141
Overflow stack (**stack guard size** configuration parameter) 11-140
Overhead
  added to procedure cache 8-9
  Component Integration Services and 8-9
  executable code 8-8
  memory 8-8
Overlapping time ranges 12-3

P

Packets, network
  size, configuring 11-75 to 11-76
page lock spinlock ratio configuration parameter 11-59
Pages, data 6-2
  allocation of 15-15, 18-2

blocksize and 21-13
dirty 11-23, 20-3
filling up transaction log 15-9, 21-31
linkage in tables and indexes 18-6, 18-11
management with extents 18-2
numbering in database 15-17
starting (*lstart*) 15-17
Pages, OAM (Object Allocation Map) 18-4
Parallel query processing
memory for 8-13
Parameters, resource limit 12-1
partition groups configuration parameter 11-120
Partitions
disk 6-3, 7-2
partition spinlock ratio configuration parameter 11-121
password expiration interval configuration parameter (now called **systemwide password expiration**) 11-135
Passwords
character sets for 14-5
null 22-6
Path name
mirror device 7-7
Percent sign (%)
error message placeholder 4-3
Performance
audit queue size 11-130
cache configuration and 9-30
database object placement and 17-4
dbcc commands 18-19
default fill factor percent effect on 11-102
disk mirroring and 5-4, 7-4
ESPs and XP Server priority 11-43
free-space accounting and 23-18
memory and 8-1, 8-2, 11-93
segments use and 17-4, 17-5
SMP environment 10-6 to 10-9
space allocation and 5-5, 17-4
speed and 5-5
windowing systems use and 8-2
permission cache entries configuration parameter 11-139
Permissions
create database 15-2
default 2-5
denying 4-8
disk init 6-6
insufficient (Level 14) 4-8
master database 2-3
mismatched *suid*s 22-15
model database 2-5
tempdb database 2-6
threshold procedures and 23-6, 23-7
transfers and 15-12
Physical resources, managing. *See* Storage management
Placeholders
error message percent sign (%) 4-3
Pointers, device. *See* Segments
Precedence
dump and **load** characteristics 21-11
resource limits 12-24
time ranges 12-25
pre-read packets configuration parameter (now called **remote server pre-read packets**) 11-79
Primary database 16-5
primary option, **disk unmirror** 7-8
print deadlock information configuration parameter 11-122
print recovery information configuration parameter 11-22, 20-7
Priority
XP Server 11-43
"proc buffers" 8-11
Procedure cache 4-11, 11-29
procedure cache percent configuration parameter and 8-3
procedure cache configuration parameter (now called **procedure cache percent**) 11-29
procedure cache percent configuration parameter 11-29

Procedures. *See* Stored procedures; System procedures
Process affinity 10-2
  engine affinity and 10-4
Processes (Server tasks) 4-14, 4-17, 10-1
  aborting when log is full 23-4
  killing 4-14 to 4-17
  suspending when log is full 23-4
Processes, SMP. *See* SMP (symmetric multiprocessing) systems
"proc headers" 8-11
Production server 3-1
Promotion, lock
  high water mark 11-110
  low water mark 11-111
  percentage 11-112
pubs2 database
  administering 2-8 to 2-9
  image information in 2-9
pubs3 database
  administering 2-8 to 2-9

Q

Queries
  conversion errors, preventing 14-4
  evaluating resource usage 12-9
  limiting resources during pre-execution and execution 12-11
  limiting with **sp_add_resource_limit** 12-1
  resource limit scope and 12-11
Query batches
  active time ranges and 12-7
  limiting elapsed time 12-15
  resource limit scope and 12-12
Question marks (??)
  for suspect characters 14-4, 14-6
Queues
  run 10-4

R

Rapid recovery 7-3
Raw devices, mirroring 7-7
read only database option 13-11, 16-6, 16-9
Reads
  physical 5-5, 7-1
Rebooting, Server. *See* Restarts, Server
Rebuilding
  master database 22-5
reconfigure command 11-18
Record keeping 3-11 to 3-13
  configuration 3-12
  contacts 3-12
  maintenance 3-12
  system 3-13
Recovery
  See also Disk mirroring
  automatic remapping 21-44
  from backups 20-17 to 20-25
  changes to user IDs 20-36
  configuration parameters for 11-22 to 11-24
  from current log 21-29
  database dump / log interactions 20-6
  default data cache and 9-35
  denying users access during 20-7
  after failures 20-6, 20-17
  failures during 21-44
  fault isolation 20-7 to 20-17
  for load option and 15-10 to 15-11
  loading databases 13-8
  master database 3-8, 6-2
  model database 22-17 to 22-18
  nonstop 5-4, 7-3
  planning backups for 2-5, 18-22
  rapid 7-3
  after reconfiguration 13-8, 13-10
  re-creating databases 21-43
  SMP engines and 10-6
  sort order changes and 13-8
  space allocation and 5-4, 21-44
  to specified time in transaction log 21-44
  step-by-step instructions 21-39 to 21-45

sybsystemprocs database 22-18 to 22-20
time and free-space accounting 23-18
time required 20-7
after upgrade 22-9
up-to-date database copy method 16-5
Recovery fault isolation 20-7 to 20-17
recovery flags configuration parameter (now called **print recovery information**) 11-22
recovery interval configuration parameter (now called **recovery interval in minutes**) 11-22
recovery interval in minutes configuration parameter 11-22 to 11-24
long-running transactions and 11-23, 20-4
Recovery of *master* database 22-2 to 22-16
automatic 20-6
backing up after 22-16
dropped users and 22-15
rebuilding 22-5
scheduling backups 20-35
user IDs and 20-36
volume changes during backup 20-37
Redundancy, full. *See* Disk mirroring
Reference information
dbcc tables 19-1
Referential integrity
memory for 8-15
Referential integrity constraints
loading databases and 21-51
reindex option, **dbcc** 13-12
remote access configuration parameter (now called **allow remote access**) 11-72
Remote backups 20-25, 20-30
remote connections configuration parameter (now called **number of remote connections**) 11-78
Remote logins
configuration parameters for 11-72
remote logins configuration parameter (now called **number of remote logins**) 11-78
Remote procedure calls
backups 20-26
memory 8-15
thresholds and 23-17
remote server pre-read packets configuration parameter 11-79
Remote servers
memory for 8-14
remote sites configuration parameter (now called **number of remote sites**) 11-79
remove option, **disk unmirror** 7-8, 7-9
Removing. *See* Dropping
Replication
recovery and 21-45
Replication Server 21-45
REPLY command (OpenVMS) 20-25, 21-34
Reporting errors 4-7, 4-10, 4-12
Reports
See also Information (Server)
dbcc 18-8, 18-14, 18-41
for **dbcc checkalloc** 18-17
for **dbcc indexalloc** 18-17
Reset configuration. *See* Configuration parameters; **reconfigure** command
Resource limits 12-1
changing 12-21 to 12-22
configuring 11-97
creating 12-17 to 12-19
dropping 12-22 to 12-24
enabling 12-2
examples of creating 12-18 to 12-19
examples of dropping 12-24
examples of getting information about 12-20
examples of modifying 12-22
identifying users and limits 12-7 to 12-13
information about 12-19 to 12-21
limiting I/O cost 12-13 to 12-15

modifying 12-21 to 12-22
planning for 12-2
precedence 12-24
pre-execution and execution 12-11
scope of 12-11
time of enforcement 12-11
understanding limit types 12-13 to 12-17
Resource usage, evaluating 12-9
Response time 11-127
Restarts, Server
after reconfiguration 13-11
automatic recovery after 20-6 to 20-7
checkpoints and 16-5
reindexing after 13-11, 13-12
from same directory 4-5
system tables and 13-11
temporary tables and 2-7
Results
limiting how many rows returned 12-16
retaindays option 21-20 to 21-22
dump database 11-25, 20-31
dump transaction 11-25, 20-31
retain option, **disk unmirror** 7-8
Return status
system procedures 1-8
Rolling back processes
recovery interval and 11-22
server stack capacity and 11-143
uncommitted transactions 20-6
roman8 character set 11-48
@@rowcount global variable
resource limits and 12-10
row count limits and 12-16
Row-offset table, checking entries in 18-11
Rows, table
limiting how many returned 12-10, 12-16
sysindexes 5-9
runnable process search count configuration parameter 11-123
Running out of space. *See* Space

S

"sa" login 22-9, 22-14
password 22-6
Savepoints
error (Level 13) 4-8
Scan descriptors 11-114 to 11-117
Scheduling, Server
database dumps 20-34
dbcc commands 18-20 to 18-22
Scope of resource limits 12-11
elapsed time 12-16
I/O cost 12-15
row count 12-17
Scripts
for backups 20-36
installdbccdb 18-37
installmaster 22-18
installmodel 22-16
logical device names in 20-32
Secondary database 16-5
secondary option, **disk unmirror** 7-8
secure default login configuration parameter 11-133
segmap column, *sysusages* table 15-17
procedures that change 5-8
segment values 21-41
Segmentation errors 22-2
segment column, *syssegments* table 15-17
Segments 5-8, 15-18 to 15-22, 17-3 to 17-25
See also Database devices; Space allocation
assigning a table or an index to specific 17-22
clustered indexes on 17-15, 17-24
creating 5-5, 17-7
creating database objects on 17-10
database object placement on 17-3, 17-10, 17-12, 21-44
default 5-5, 17-2
dropping 17-16
extending 17-8
free-space accounting and 23-18

information on 15-18 to 15-22, 17-16 to 17-18, 18-18
listing thresholds for 23-5
logsegment 5-5, 17-2, 23-1 to 23-18
managing free space 23-1 to 23-18
nonclustered indexes on 17-12
performance enhancement and 17-4, 17-5
placing objects on 17-3, 17-12, 21-44
removing devices from 17-16
sharing space on 17-3
sp_helpthreshold report on 23-5
syssegments table 5-8
system segment 5-5, 17-2
system tables entries for 15-17, 17-18 to 17-19
text/image columns and 17-15
thresholds and 23-11
tutorial for creating 17-19 to 17-24
user-defined 21-41
values table 15-17
select into/bulkcopy/pllsort database option
model database and 2-5
transaction log and 16-6
select into command
database dumping and 20-35
select on syscomments.text column configuration parameter 11-134
Separation, physical
of transaction log device 7-2
serial option, **disk mirror** 7-7
server.loc file 13-4
server_name.cfg, default name of configuration file 11-6
Server engine. *See* Engines
Server information options. *See* Information (Server)
Servers
architecture for SMP 10-2 to 10-4
connecting 1-11
database creation steps 15-4
error messages 4-4
error message severity levels 4-6 to 4-12
fatal errors and 4-10 to 4-12
installing 3-3 to 3-4, 5-5
interfaces files 1-11
master-recover mode 22-6
memory needs 8-1, 11-29, 11-92
monitoring performance 11-18
multiprocessor 10-1 to 10-9
nonfatal internal errors 4-10
object placement on segments 17-12, 21-44
scheduler 11-110, 11-127
shutting down 4-23
single-user mode 11-99, 16-7, 22-3, 22-6
sort order consistency among 13-8
space allocation steps 15-15
start-up problems and memory 8-2, 11-93
syntax errors 4-8, 4-9
task management in SMP 10-4
user connections to 11-139
values for configuration parameters 11-5
Sessions. *See* Logins
set command
char_convert 14-6 to 14-7
7-bit ASCII character data, character set conversion for 13-8, 14-1, 14-5
Severity levels, error 4-1, 4-6
Backup Server 4-13
levels 10-18 (user errors) 4-7
levels 19-24 (fatal) 4-10
shared memory starting address configuration parameter 11-84
showplan option, **set**
resource limits and 12-9, 12-10
showserver utility command 22-13
See also Utility Programs manual
shutdown command 4-23 to 4-25
automatic checkpoint process and 20-6
automatic recovery after 20-6
Backup Server 20-30
Shutting down servers 4-23

side option, **disk unmirror** 7-8
Single-byte character sets 13-13
single user database option 16-7
Single-user mode 11-99, 13-11, 22-3
Size
  See also Space
  allocation units 6-2, 15-15
  altering database 15-12 to 15-14
  database 15-5
  database device 6-4
  databases, estimating 15-7
  dbcc fix_text transaction 13-13
  error log 1-10
  indexes 15-7
  memory 8-1 to 8-11, 11-92
  model database 6-4, 11-101, 15-6
  new database 2-4, 15-6
  segment extension 17-8
  tables 15-7
  tape dump device 20-33
  tempdb database 2-6
  text storage 15-21
  transaction logs 15-8, 16-7
size of auto identity column configuration parameter 11-124, 16-3
  unique auto_identity index database option and 16-8
size option
  disk init 6-4
sjis (Shift-JIS) character set. *See* Japanese character sets
Sleeping checkpoint process. *See* Checkpoint process
Sleep queue 10-4
SMP (symmetric multiprocessing) systems
  architecture 10-2 to 10-4
  environment configuration 10-5 to 10-9
  managing Servers on 10-1 to 10-9
  task management in 10-4 to 10-5
Sort order
  changing 13-8 to 13-10, 13-12, 18-13
  consistency among servers 13-8
  database dumps and 21-26
  dbcc checktable and 18-12
  default sortorder id 11-49
  definition files 13-2
  information about 13-8
  installing new 13-3
  numbers 11-49
  rebuilding indexes after changing 13-10 to 13-14
sp_add_resource_limit system procedure 12-17
sp_add_time_range system procedure 12-4
sp_addlanguage system procedure 13-15
sp_addlogin system procedure
  reissuing after recovery 22-15
sp_addsegment system procedure 17-7, 17-19
  sysusages and 5-8
sp_addthreshold system procedure 23-6 to 23-11
sp_addumpdevice system procedure 20-33
sp_adduser system procedure 2-5, 15-5
sp_cacheconfig system procedure 9-4 to 9-10
sp_changedbowner system procedure 15-2, 15-12
sp_configure system procedure 11-8
  See also individual configuration parameter names
  automatic recovery and 20-3
sp_countmetadata system procedure 11-63, 11-65, 11-67
sp_dboption system procedure 16-1 to 16-9
  aborting processes 23-4
  changing default settings with 15-4
  checkpoints and 20-6
  disabling free-space accounting 23-18
  disk unmirroring and 7-10
  thresholds and 23-4
sp_diskdefault system procedure 5-2, 6-7 to 6-9
sp_drop_resource_limit system procedure 12-22

sp_drop_time_range system procedure 12-6
sp_dropalias system procedure 15-12
sp_dropdevice system procedure 6-7, 15-15, 20-33
  for failed devices 21-42
sp_droplogin system procedure
  reissuing after recovery 22-15
sp_dropsegment system procedure 17-16
  sysusages and 5-8
sp_dropthreshold system procedure 23-7
sp_dropuser system procedure 15-12
sp_estspace system procedure 15-7
sp_extendsegment system procedure 17-8
  reversing effects of 17-9
  sysusages and 5-8
sp_forceonline_db system procedure 20-11 to 20-12
sp_forceonline_page system procedure 20-11 to 20-12
sp_help_resource_limit system procedure 12-19, 12-20
sp_helpcache system procedure 9-17
sp_helpconfig system procedure 8-6, 11-62, 11-64, 11-67
sp_helpdb system procedure 1-8
  database option information 16-2
  segment information 17-18
  storage information 15-18
sp_helpdevice system procedure 1-8, 6-6, 20-32
sp_helpindex system procedure 1-8
sp_helpjoins system procedure 1-6
sp_helpkey system procedure 1-6
sp_helplog system procedure 15-10
sp_helpsegment system procedure 17-17, 17-18
  checking space with 20-3
sp_helpsort system procedure 13-8
sp_helptext system procedure 1-8
sp_helpthreshold system procedure 23-5
sp_indsuspect system procedure 13-11, 13-12
sp_listsuspect_db system procedure 20-10
sp_listsuspect_page system procedure 20-11
sp_locklogin system procedure
  reissuing after recovery 22-15
sp_logdevice system procedure 15-9 to 15-10, 17-5
sp_modify_resource_limit system procedure 12-21
sp_modify_time_range system procedure 12-6
sp_modifylogin system procedure 13-10
sp_modifythreshold system procedure 23-6
sp_monitorconfig system procedure 8-7
  configuring **number of aux scan descriptors** and 11-116
  configuring **number of open databases** and 11-63
  configuring **number of open indexes** and 11-66
  configuring **number of open objects** and 11-68
sp_placeobject system procedure 17-12, 17-13
sp_reportstats system procedure
  resource limits and 12-8
sp_setsuspect_granularity system procedure 20-9 to 20-10
sp_setsuspect_threshold system procedure 20-9 to 20-10
sp_showplan system procedure 4-22
sp_spaceused system procedure 15-19
  checking transaction logs with 20-3
sp_sysmon system procedure
  wash size and 9-22, 9-23
sp_thresholdaction system procedure 23-2
  creating 23-12 to 23-18
  dumping transaction log 23-13
  error messages and 23-13
  parameters passed to 23-13
  sample procedure 23-14 to 23-17
sp_volchanged system procedure 21-34
sp_who system procedure
  checkpoint process 20-4
  LOG SUSPEND status 23-5

Space
See also Size; Space allocation
adding to database 15-12 to 15-14
estimating table and index size 15-7
extending database 15-12 to 15-14
information on usage 15-19, 21-41
proportion of log to database 15-8
reserved 15-20
running out of 4-10, 16-7, 21-31
sharing on segments 17-3
sp_dropsegment effect on 17-16
between thresholds 23-11 to 23-12
unreserved 15-20
Space allocation
See also Database devices; Segments; Storage management
assigning 15-5, 21-43
backup methods and 21-41
balance and split tables 17-5
changing once assigned 15-7, 15-12, 17-12
commands summary 5-2
contiguous 6-5, 15-15, 15-17
dbcc commands for checking 18-13 to 18-15
disk mirroring and 7-1 to 7-2
drop database effect on 15-14
error correction with **dbcc** 18-14, 18-17
on an existing device 21-43
extents 18-2
extents and **sp_spaceused** report 15-20
fixing unreferenced extents 18-17
functions of Server 15-15, 18-2
matching new database to existing 21-41
Object Allocation Map (OAM) 18-4
pages 15-20, 17-12, 18-2
recovery / performance and 5-4 to 5-5, 17-4
re-creating 20-19, 21-44
segments and 21-44
sysusages table 5-8
units 6-2, 15-15, 18-2, 21-41
#spdevtab temporary table 1-8
Speed (Server)
of **dbcc** commands 18-19
system performance and 5-5, 7-4
of transaction log growth 15-8
using segments 17-1
#spindtab temporary table 1-8
Spinlocks
configuration parameters affecting 10-9
Splitting
tables across segments 17-12 to 17-13
tables across two disks 5-5
SPR files 4-25
spt_committab table 1-8
spt_limit_types table 12-10
spt_monitor table 1-8
spt_values table 1-8
sql server clock tick length configuration parameter 11-126
sql server code size configuration parameter (now called **executable code size**) 11-61
Square brackets [ ]
in SQL statements xxxvii
.srt files 13-2
stack guard size configuration parameter 11-140
stack size configuration parameter 11-143
Standalone utilities and character sets 14-5
Starting Servers
Backup Server 20-30
master-recover mode 22-6
memory required for 8-2
start mail session configuration parameter 11-45
startserver utility command
Backup Server and 20-30
master-recover mode 22-6
Static configuration parameters 11-6
Statistics
backup and recovery 20-39
dbcc output 18-24
I / O cost 12-14

statistics io option, **set**
resource limits and 12-9, 12-10
statistics time option, **set**
determining processing time 12-15
resource limits and 12-9, 12-10
Status
information messages (Level 10) 4-7
status bits in *sysdevices* 6-6
Stopping
Backup Server 4-24, 20-30
Servers 4-24
Storage management 5-1
See also Space; Space allocation
changing database ownership 15-12
commands summary 5-2
creating user databases 15-3 to 15-12
database device initialization 6-1 to 6-7
default database devices 6-7 to 6-9, 17-8
defaults at installation 5-5
disk mirroring 7-1 to 7-10
dropping databases 15-14
hints 3-6
information about 15-20
issues 3-4 to 3-7, 5-4 to 5-5, 17-4
system tables and 5-6 to 5-9
using segments 17-3 to 17-25
Stored procedures
cache binding and 9-30
creating 1-9
procedure cache and 11-29
resource limit scope and 12-11
system tables changes and 1-9
stripe on option 21-17 to 21-20
Structure
configuration 10-5 to 10-9
internationalization files directory 13-3
localization files directory 13-5
Suffix names, temporary table 2-6
Sun character set 11-48
Superuser. *See* System Administrator
Suspect escalation threshold 20-9
Suspect pages
assessing 20-16
isolating on recovery 20-7 to 20-17
listing 20-10
suspend audit when device full configuration parameter 11-134
Sybase Central, using for system administration tasks 1-3
sybsecurity database 2-7
automatic recovery and 20-6
sybsyntax database 2-9
sybsystemdb database 2-8
automatic recovery and 20-6
sybsystemprocs database 1-7, 1-9, 2-5 to 2-6
See also Databases
automatic recovery and 20-6
backing up 20-38
restoring 22-18 to 22-20
thresholds and 23-17
Symbols
See also Symbols section of this index
in SQL statements xxxvi
Symmetric multiprocessing systems. *See* SMP (symmetric multiprocessing) systems
Syntax
errors in 4-8, 4-9
Transact-SQL conventions xxxvi to xxxix
syscolumns table 18-18
sysconfigures table 11-20
syscurconfigs table 11-20
sysdatabases table
create database and 15-4
disk refit and 22-22
sysdevices table 5-7 to 5-8, 6-6
create database and 15-6
disk init and 5-8
disk mirroring commands and 7-6
dump devices and 20-32
sp_dropdevice and 6-7
sp_helpdevice and 6-6
status bits 6-7, 15-18

sysindexes table 5-9, 13-11, 17-15
syslogins table
backup and recovery 20-36
character set conversion and 14-4
resource limits and 12-9
syslogs table 20-2
See also Transaction logs
create database and 15-3, 15-7
modification of 1-7
monitoring space used by 15-21
put on a separate device 7-2
sysmessages table 4-2, 4-3
sysobjects table 13-11
sysprocesses table
resource limits and 12-8
sysresourcelimits table 12-19
syssegments table 5-8, 15-17, 17-18
sysservers table
Backup Server and 20-28
System administration tasks
accomplishing with Sybase Central 1-3
System Administrator 1-1 to 1-2
error responsibilities of 4-7, 4-9 to 4-12
password and **buildmaster** 22-6
resolving system problems 4-7, 4-9
single-user mode of Server 22-6
tasks for beginners 3-1 to 3-13
System catalogs. *See* System tables
System databases 2-1 to 2-9
System extended stored procedures 1-9
System messages. *See* Error messages; Messages
System problems
See also Errors
Server responses to 4-1 to 4-12
severity levels 10 to 18 4-7 to 4-10
severity levels 19 to 24 4-10 to 4-12
System Problem Reports (SPRs) 4-25
System procedures 1-7 to 1-9
See also Information (Server); Stored procedures; *individual procedure names*
creating 1-9
on temporary tables 2-7
using 1-7
for using segments 17-3
System procedure tables 1-8
System roles
max roles enabled per user configuration parameter 11-132
system segment 5-5, 17-2
cannot entirely drop from a database 17-16
System tables 1-4 to 1-6
See also Tables; *individual table names*
changes dangerous to 1-9
corruption 4-12
create database and 1-5, 5-8, 17-19
creation of 1-5
dbcc checkcatalog and 18-18
dbcc nofix option 18-16
dbcc reindex and 13-13
direct updates dangerous to 22-7
keys for 1-6
querying 1-5, 1-9
reindexing and 13-13
segment information and 17-18 to 17-19
Server restarts and 13-11
storage management relationships 5-6 to 5-9
stored procedures and 1-5, 1-9
updating 1-6, 1-9, 22-7
for user databases 2-4
systemwide password expiration configuration parameter 11-135
systhresholds table 23-17
systimeranges table
dropping time ranges 12-6
range IDs 12-4
sysusages table 5-8, 17-18
corruption 4-12
create database and 15-4, 21-43
database space allocations and 15-16, 21-41
discrepancies in 22-16

disk refit and 22-21 to 22-22
recovery and 22-7

T

T1204 trace flag (now called **print deadlock information** configuration parameter) 11-122
T1603 trace flag (now called **allow sql server async i/o** configuration parameter) 11-36
T1610 trace flag (now called **tcp no delay** configuration parameter) 11-80
T1611 trace flag (now called **lock shared memory** configuration parameter) 11-91
tablealloc option, **dbcc** 18-16, 18-19
table lock spinlock ratio configuration parameter 11-60
Tables
See also Database objects; System tables
assigning to specific segments 17-22
binding to data caches 9-15
critical data in 18-22
dbcc checkdb and 18-13, 18-19
dbcc checktable and 13-12, 15-9, 15-10, 18-11, 18-19
integrity checking with **dbcc** 18-11
integrity damage to 4-11
migration to a clustered index 17-15, 17-24
moving between devices 17-12, 17-15, 17-24
Object Allocation Maps of 11-28, 18-4
read-only 13-11
sort order of 18-12
splitting across segments 17-12 to 17-13
splitting across two disks 5-5
system procedure 1-8
temporary 2-5 to 2-6
without indexes 13-12
Tape dump devices
adding 20-33
for backups 20-31
dismounting 21-21
end-of-tape marker 21-13
preventing overwrites 20-31, 21-22
reinitializing volumes 21-23
rewinding 21-21, 21-22
volume name 21-14
Tape labels
information on dump files 20-22
tape retention configuration parameter (now called **tape retention in days**) 11-25
tape retention in days configuration parameter 11-25, 20-31
Tasks
SMP task management 10-4
tcp no delay configuration parameter 11-80
tempdb database 2-6
See also Databases
auto identity database option and 16-3
automatic recovery and 20-6
creating 5-5
data caches 9-36
size of 2-6
unique auto_identity index database option and 16-8
Temporary tables 2-5 to 2-6
select into/bulkcopy/pllsort database option and 16-6
Terminals
character set conversion for 14-7
installing new definitions 13-3
Test servers 3-1 to 3-3
text datatype
chain of text pages 17-15
changing character sets and 13-13
multibyte character sets and 13-13
performance effects of 17-6
size of storage 15-21
storage on separate device 17-15
sysindexes table and 17-15, 17-19

text values, **dbcc fix_text** upgrade of 13-13, 13-13 to 13-14
@@thresh_hysteresis global variable 23-3
  threshold placement and 23-11
Threshold procedures
  creating 23-12 to 23-18
  creating, logical names and 20-32
  dumping transaction log and 23-13
  error messages and 23-13
  location of 23-6, 23-17
  parameters passed to 23-13
  permissions for 23-6, 23-7
Thresholds 23-1 to 23-18
  adding 23-6 to 23-11
  adding for log segment 23-8 to 23-11
  changing 23-6
  creating 23-5 to 23-11
  disabling 23-18
  finding associated procedure 23-17
  hysteresis value 23-3
  information about 23-5
  last-chance 23-1 to 23-18
  maximum number 23-5
  midpoint between two 23-11
  removing 23-7
  segments and 23-11
  space between 23-11 to 23-12
  systhresholds table 23-17
Time interval
  database backups 20-34
  limiting 12-10
Time ranges 12-3
  adding 12-4 to 12-5
  "at all times" 12-3
  changing active time ranges 12-7
  creating 12-4 to 12-5
  dropping 12-6
  dropping resource limits using 12-23
  modifying 12-6
  overlapping 12-3
  precedence 12-25
  using 12-3 to 12-7
time slice configuration parameter 11-127
Timestamps, order of transaction log dumps 21-44
Time values
  display format 13-4
Timing
  automatic checkpoint 20-3
total data cache size configuration parameter 11-30
total memory configuration parameter 8-2, 10-8, 11-92
Transaction logs
  See also Dump, transaction log; **dump transaction** command; *syslogs* table
  alter database and 5-8
  backing up 20-18
  caches and 9-10
  checking space used by 15-8
  clearing after checkpoints 20-4 to 20-5
  copying 20-3
  create database and 5-8, 15-7
  data caches and 9-10
  device placement 5-4, 5-8, 15-7, 15-10, 15-11
  dumping after media failure 21-28
  function of 20-2
  master database 20-37
  model database 20-37
  modifying between loads 21-44
  moving to release space 15-10
  primary and secondary database 16-5
  purging 13-14, 21-31
  room for growth 20-3
  running out of space 20-22
  on same device 20-22, 21-30
  select into/bulkcopy/pllsort database option 16-6
  on a separate device 7-2, 20-18
  size 15-8, 16-7, 20-3
  synchronizing with database 20-3 to 20-6
  thresholds and 23-14
  truncating 21-30 to 21-32
  trunc log on chkpt option and 11-23, 16-7
  unlogged commands 20-35

Transactions
See also Locks; Transaction logs
active time ranges and 12-7
definition 20-2
error within 4-8
limiting elapsed time 12-15
limiting with **sp_add_resource_limit** 12-1
long-running 11-23, 20-4
recovery and 11-23, 20-4
resource limit scope and 12-12
two-phase commit 2-8
Translation. *See* Character sets
Triggers
nested 11-97
truncate_only option, **dump transaction** 21-30
trunc log on chkpt database option 16-7
recovery interval in minutes and 11-23
Tuning
monitoring performance 11-18
Tutorial for creating segments 17-19 to 17-24
Two-phase commit
transactions 2-8

U

unique auto_identity index database option 16-8
UNIX platforms, raw disk partition 6-3
unload option 21-20 to 21-22
Unlogged commands 20-35
Unmirroring devices. *See* Disk mirroring
update command
transaction log and 15-8, 20-3
Updating
See also Changing
allow updates to system tables configuration parameter and 1-9
current transaction log page 15-10
system tables 22-7
text after character set change 13-13
Upgrade, recovery after 22-9
upgrade version configuration parameter 11-128
us_english language 11-48, 14-5, 14-7
User connections
memory allocated per 11-137 to 11-139
user connections configuration parameter (now called **number of user connections**) 11-137
User databases
automatic recovery and 20-6
creation process 15-4
master database control of 2-2
system tables for 2-4
user-defined characters (Gaiji) 14-3
user-defined messages 4-6
User-defined roles
number allowed 11-132
User errors 4-7, 4-7 to 4-10
User IDs
comparing after backup and recovery 20-36, 22-15
number 1, Database Owner 1-9
user log cache size configuration parameter 11-144
user log cache spinlock ratio configuration parameter 11-145
User mistakes. *See* Errors; Severity levels, error
User names
character set conversions and 14-5
Users
added, and recovery of *master* 22-15
adding to databases 15-5
dropped, and recovery of *master* 22-15
dropping from databases 15-5
dropping resource limits on 12-23
errors by 4-7, 4-7 to 4-10
getting resource limit information about 12-19
identifying usage-heavy 12-8
modifying resource limits on 12-21
multiple, and performance 17-12
names of 14-5
number of user connections and 11-138

single-user mode 11-99, 16-7
User segments, creating 17-19 to 17-24
See also Segments
use security services configuration parameter 11-137
Utility commands
See also Utility Programs manual
buildmaster 22-6
character sets and 14-5
showserver 22-13
startserver 22-6

V

Variables
in error messages 4-3
vdevno option
disk init 6-3
Version identifiers, automatic upgrade and 21-49
Virtual address 6-5
Virtual device number 15-18
Virtual page numbers 6-5
Virtual Server Architecture 10-1
Volume handling 21-14
vstart column 15-18
vstart option
disk init 6-5

W

waitfor mirrorexit command 7-10
Wash area
configuring 9-20 to 9-23
defaults 9-21
Windowing systems 8-2
Window of vulnerability 11-98
with no_error option, **set char_convert** 14-6
with no_log option, **dump transaction** 21-31
with no_truncate option, **dump transaction** 21-28 to 21-30
with nowait option, **shutdown** 4-24, 4-25
with override option
create database 15-11
with truncate_only option, **dump transaction** 21-30
Write-ahead log. *See* Transaction logs
Write operations
disk mirroring and 7-1
physical 5-5
writes option, **disk mirror** 7-7
writetext command
database dumping and 20-35
select into/bulkcopy/pllsort database option 16-6

X

.xlt files 13-2
xp_cmdshell context configuration parameter 11-46
xp_cmdshell system extended stored procedure 1-9
XP Server
freeing memory from 11-44
priority 11-43