SYBASE®

System Administration Guide: Volume 2

Adaptive Server® Enterprise

15.0

DOCUMENT ID: DC31644-01-1500-02

LAST REVISED: October 2005

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

This manual, the*System Administration Guide: Volume 2*, describes how to administer and control Sybase® Adaptive Server® Enterprise databases independent of any specific database application.

Audience

This manual is for Sybase System Administrators and Database Owners.

How to use this book

This manual contains the following chapters:

- Chapter 1, "Limiting Access to Server Resources," explains how to create and manage resource limits with Adaptive Server.
- Chapter 2, "Mirroring Database Devices," describes how to mirror database devices for nonstop recovery from media failures.
- Chapter 3, "Configuring Memory," explains how to configure Adaptive Server to use the available memory on your system.
- Chapter 4, "Configuring Data Caches," discusses how to create named caches in memory and bind objects to those caches.
- Chapter 5, "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.
- Chapter 6, "Creating and Managing User Databases," discusses the physical placement of databases, tables, and indexes, and the allocation of space to them.
- Chapter 7, "Database Mount and Unmount," describes how to transport databases from a source Adaptive Server to a destination Adaptive Server.
- Chapter 8, "Creating and Using Segments," describes how to use segments, which are named collections of database devices, in databases.
- Chapter 9, "Using the reorg Command," describes how to use the reorg command.

- Chapter 10, "Checking Database Consistency," describes how to use the database consistency checker, dbcc, to detect and fix database problems.
- Chapter 11, "Developing a Backup and Recovery Plan," discusses the capabilities of the Backup Server and how to develop your backup strategy.
- Chapter 12, "Backing Up and Restoring User Databases," discusses how to recover user databases.
- Chapter 13, "Restoring the System Databases," discusses how to recover system databases.
- Chapter 14, "Automatic Database Expansion," describes how to configure databases to expand automatically when they run out of space.
- Chapter 15, "Managing Free Space with Thresholds," discusses managing space with thresholds.

Volume 1 of the *System Administration Guide* contains the following chapters:

- Chapter 1, "Overview of System Administration," describes the structure of the Sybase system.
- Chapter 2, "System and Optional 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 must perform.
- Chapter 4, "Introduction to the Adaptive Server Plug-in for Sybase Central," – describes how to start and use Sybase Central, a graphical user interface for managing Adaptive Server.
- Chapter 5, "Setting Configuration Parameters," summarizes the configuration parameters that you set with sp_configure, which control many aspects of Adaptive Server behavior.
- Chapter 6, "Overview of Disk Resource Issues," discusses Adaptive Server and Backup Server™ error handling and how to shut down servers and kill user processes.
- Chapter 7, "Initializing Database Devices," describes how to initialize database devices and assign devices to the default pool of devices.
- Chapter 8, "Setting Database Options," describes how to set database options.

- Chapter 9, "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 10, "Configuring Client/Server Character Set Conversions," discusses character set conversion between Adaptive Server and clients in a heterogeneous environment.
- Chapter 11, "Diagnosing System Problems," discusses Adaptive Server and Backup Server error handling and shows how to shut down servers and kill user processes.
- Chapter 12, "Introduction to Security," introduces you to security concepts.
- Chapter 13, "Getting Started With Security Administration in Adaptive Server," provides an overview of the security features available in Adaptive Server.
- Chapter 14, "Managing Adaptive Server Logins, Database Users, and Client Connections," describes methods for managing Adaptive Server login accounts and database users.
- Chapter 15, "Managing Remote Servers," discusses the steps the System Administrator and System Security Officer of each Adaptive Server must execute to enable remote procedure calls (RPCs).
- Chapter 16, "External Authentication," describes the network-based security services that enable you to authenticate users and protect data transmitted among machines on a network.
- Chapter 17, "Managing User Permissions," describes the use and implementation of user permissions.
- Chapter 18, "Auditing," describes how to set up auditing for your installation.
- Chapter 19, "Confidentiality of Data," describes how to configure Adaptive Server to ensure that all data is secure and confidential.

Related documents

The Sybase Adaptive Server Enterprise documentation set consists of the following:

- 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 the Sybase Technical Library.

- The *Installation Guide* for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 15.0, the system changes added to support those features, and changes that may affect your existing applications.
- *ASE Replicator User's Guide* – describes how to use the Adaptive Server Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.
- *Component Integration Services User's Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- The *Configuration Guide* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.
- *Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server User's Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as data types, functions, and stored procedures in the Adaptive Server database.
- *Job Scheduler User's Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Messaging Service User's Guide* – describes how to useReal Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.

- *Monitor Client Library Programmer's Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server User's Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Guide* – is a series of four books for Adaptive Server version 12.5.x that explains how to tune Adaptive Server for maximum performance:
  - *Basics* – the basics for understanding and investigating performance questions in Adaptive Server.
  - *Locking* – describes how the various locking schemas can be used for improving performance in Adaptive Server.
  - *Optimizer and Abstract Plans* – describes how the optimizer processes queries and how abstract plans can be used to change some of the optimizer plans.
  - *Monitoring and Analyzing* – explains how statistics are obtained and used for monitoring and optimizing performance.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book.
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL® information:
  - *Building Blocks* – Transact-SQL datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
  - *Commands* – Transact-SQL commands.
  - *Procedures* – Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and `dbcc` stored procedures.
  - *Tables* – Transact-SQL system tables and `dbcc` tables.
- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.

- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
- *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.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase's Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – Describes the Unified Agent, which provides runtime services to manage, monitor and control distributed Sybase resources.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services User's Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.
- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that comprise XML Services.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.

- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
3. Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBFs/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBFs/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBFs/Maintenance report, or click the product description to download the software.

Conventions

This section describes 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.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Adaptive Server documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

SQL syntax conventions

Table 1 lists the conventions for syntax statements in this manual:

Table 1: Syntax statement conventions

Key	Definition
`command`	Command names, command option names, utility names, utility flags, and other keywords are in `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 indicate that choosing one or more of the enclosed options is optional. Do not include brackets in your option.
`( )`	Type parentheses 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.

- 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 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

- One item in square brackets: You do not 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

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 may also be used inline to signify portions of a command that are omitted from 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”
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

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Adaptive Server documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

CHAPTER 1

Limiting Access to Server Resources

This chapter describes how to use resource limits to restrict the I/O cost, row count, processing time, or tempdb space 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.

What are resource limits?

Adaptive Server provides resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. Resource limits, however, are not fully specified until they are bound to a time-range.

A *resource limit* is a set of parameters specified by a System Administrator to prevent an individual login or application from exceeeding the limits described in Table 1-1.

Table 1-1: Resource limits

LImit being tracked	What is tracked by Adaptive Server
Exceeding I/O costs	Estimated costs – as determined by the optimizer. Actual costs – as measured during query execution
Returning excessive number of rows	Tracked on a per-query basis
Exceeding a given elapsed time	On a query batch basis in a given transaction.
Utilizing excessive tempdb space	Tracked per session

Resource limits are bound to time-ranges, allowing the System Administrator to define precisely when they should be enforced.

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 (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:

- sp_add_time_range to create a named time range to specify times for imposing the limit.
- sp_add_resource_limit to create new resource limits.
- sp_help_resource_limit to obtain information about existing resource limits.
- sp_modify_time_range and sp_modify_resource_limit to modify time ranges and resource limits, respectively.
- sp_drop_time_range and sp_drop_resource_limit to drop time ranges and resource limits, respectively.

Enabling resource limits

To configure Adaptive Server to enable resource limits, use the allow resource limits configuration parameter:

```
sp_configure "allow resource limits", 1
```

1 enables resource limits; 0 disables them. allow resource limits is static, so you must restart the server to reset the changes.

allow resource limits signals the server to allocate internal memory for time ranges, resource limits, and internal server alarms. It also 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. 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 "Messages describing access methods, caching, and I/O cost" on page 93 in the *Performance and Tuning Guide: Monitoring and Analyzing*.

- statistics i/o includes the actual total I/O cost of a statement according to the optimizer's costing formula. 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.

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.

Adaptive Server includes a predefined "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 cannot 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 see 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, you can put 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 refer to specific time ranges.

Determining the time ranges you need

Use a chart like the one below to determine the 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 use sp_add_time_range 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, see sp_add_time_range in the *Reference Manual*.

A time range example

Assume that two critical jobs 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

Use sp_modify_time_range 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, see sp_modify_time_range in the *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, enter:

```
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, enter:

```
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

Use sp_drop_time_range to drop a user-defined time range.

For syntax and detailed information, see sp_drop_time_range in the *Reference Manual*.

For example, to remove the *evenings* time range from the systimeranges system table in the master database, enter:

```
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 is affected by the change in time.

Adding, modifying, and deleting time ranges does not affect the active time ranges for the login sessions 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 sp_reportstats. The output from this procedure can help you identify 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 “Setting Configuration Parameters” on page 61 and “Setting Configuration Parameters” on page 61.

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 3.

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.

Choosing a limit type

After you determine the users and applications to limit, you have a choice of three different types of resource limits.

Table 1-2 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.

Table 1-2: 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	Preexecution 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
tempdb_space	Use all space in tempdb when creating work or temporary tables.	Number of pages used in tempdb per session.	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:

- Preexecution – 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 preexecution 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 is actually executed.

A preexecution 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 preexecution time resource limits statements in a trigger.

- Execution – Adaptive Server applies resource limits at runtime, and is usually used to prevent a query from monopolizing server and operating system 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 preexecution 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 – 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 four types of resource limits that allow you to limit resource usage in different ways:

- Limiting I/O cost
- Identifying I/O costs
- Calculating the I/O cost of a cursor
- scoping of the io_cost limit type

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).

To set resource limits, you must understand how those limits translate into runtime 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 exceeds 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 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 preexecution time resource limits. A preexecution 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 needed requires a physical I/O for the first access and is 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 is 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.

The optimizer's estimates are not 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, which is calculated when the cursor opens.

Preexecution 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 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 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.

To help develop appropriate limits for elapsed time use information you have gathered with set statistics 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. Convert 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 continues 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 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 preexecution 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 11.

Setting limits for tempdb space usage

The tempdb_space resource limit restricts the number of pages a tempdb database can have during a single session. If a user exceeds the specified limit, the session can be terminated, or the batch or transaction aborted.

For queries executed in parallel, the tempdb_space resource limit is distributed equally among the parallel threads. For example, if the tempdb_space resource limit is set at 1500 pages and a user executes the following with three-way parallelism, each parallel thread can create a maximum of 500 pages in tempdb:

```
select into #temptable from partitioned_table
```

The system administrator or database administrator sets the tempdb_space limit using sp_add_resource_limit, and drops the tempdb_space limit using sp_drop_resource_limit.

Creating a resource limit

Create a new resource limit with sp_add_resource_limit. The syntax 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 type of limit (io_cost, row_count, elapsed_time, or tempdb_space), and set an appropriate value for the limit type.

  For more information, see "Choosing a limit type" on page 9.

- Specify whether the resource limit is enforced prior to or during query execution.

  Specify numeric values for this parameter. Preexecution 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 10.

- 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 11.

For detailed information, see sp_add_resource_limit in the *Reference Manual*.

Resource limit examples

This section includes three examples of setting resource limits.

Examples

Example 1 This example creates a resource limit that applies to all users of the payroll application because the name parameter is NULL:

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 120, 2, 1, 2
```

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 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:

```
sp_add_resource_limit joe_user, NULL, saturday_night,
    row_count, 5000, 2, 3, 1
```

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 This example also creates a resource limit that applies to all ad hoc queries and applications run by "joe_user":

```
sp_add_resource_limit joe_user, NULL, "at all times",
    io_cost, 650, 1, 3, 1
```

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 preexecution time (*enforced* = 1).

Note Although Adaptive Server terminates the current transaction when it reaches its time limit, you receive no 1105 error message until you issue another SQL command or batch; in other words, the message appears only when you attempt to use the connection again.

Getting information on existing limits

Use sp_help_resource_limit to get information about existing resource limits.

Users who do not have the System Administrator role can use sp_help_resource_limit to list only their own resource limits.

Users either specify their own login names as a parameter or specify the *name* parameter as “null.” The following examples 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 display 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, see sp_help_resource_limit in the *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 1-3 shows the meaning of the values in the limitid, enforced, action, and scope columns.

Table 1-3: 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 – Before 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
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 "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 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 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 *appname* 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.
- You must specify non-null values for *rangename* and *limittype*. To uniquely identify the limit, specify non-null values for *action* and *scope*.
- Specifying "null" for *limitvalue* or *action* indicates that its value does not change.

For detailed information, see sp_modify_resource_limit in the *Reference Manual*.

Examples of modifying a resource limit

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 NULL, payroll, tu_wed_7_10,
elapsed_time, 90, null, null, 2
```

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 to 2, which aborts the query batch. The values for limit type, time of execution, and scope remain unchanged.

```
sp_modify_resource_limit joe_user, NULL,
saturday_night, row_count, NULL, NULL, 2, NULL
```

Adaptive Server Enterprise provides resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. Resource limits, however, are not fully specified until they are bound to a time range.

A resource limit is a set of parameters specified by a System Administrator to prevent an individual login or application from:

- Exceeding a given estimated or actual I/O costs.
- Returning excessive rows on a per query basis.
- Exceeding a given elapsed time on a query batch or transaction basis.
- Utilizing excessive tempdb space per session.

In Adaptive Server Enterprise version 12.5.3, when the System Administrator modifies a resource limit, all users logged in the session see the change, including the System Administrator.

See "Getting information on existing limits" on page 19 for more information.

Dropping resource limits

Use sp_drop_resource_limit to drop a resource limit from an Adaptive Server.

The syntax is:

sp_drop_resource_limit {*name* , *appname* } [, *rangename*, *limittype*, *enforced*, *action*, *scope*]

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 1-4.

Table 1-4: 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 *scope*, 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 *scope*, 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 *scope*, 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 *scope*, without regard to *action*.
scope	• One of the scope types: query, query batch, transaction	Drops the limits that apply to a particular scope.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *limittype*, enforcement time, and *action*, without regard to *scope*.

When you use sp_droplogin to drop an Adaptive Server login, all resource limits associated with that login are also dropped.

For detailed information, see sp_drop_resource_limit in the *Reference Manual*.

Examples of dropping a resource limit

Example 1 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
```

Example 2 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 NULL, payroll, tu_wed_7_10
```

Example 3 Drops all resource limits for "joe_user" from the payroll application:

```
sp_drop_resource_limit joe_user, payroll
```

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.

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.

CHAPTER 2

Mirroring Database Devices

This chapter describes how to create and administer disk mirrors.

What is disk mirroring?

Disk mirroring can provide nonstop recovery in the event of media failure. The disk mirror command causes an Adaptive Server database device to be duplicated, that is, all writes to the device are copied to a separate physical device. If one device 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.

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—only the transaction logs, all devices on a server, or selected devices.

Note You cannot mirror a dump device.

Mirror all default database devices so that you are 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 use alter database to add a second device and then run sp_logdevice.

Here are three examples that involve different cost and performance trade-offs:

- 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.
- Storage space – immediate recovery requires full redundancy (all databases and logs mirrored), which consumes disk space.
- Impact on performance – Mirroring the user databases (as shown in Figure 2-2 on page 29 and Figure 2-3 on page 30) increases the time needed to write transactions to both disks.

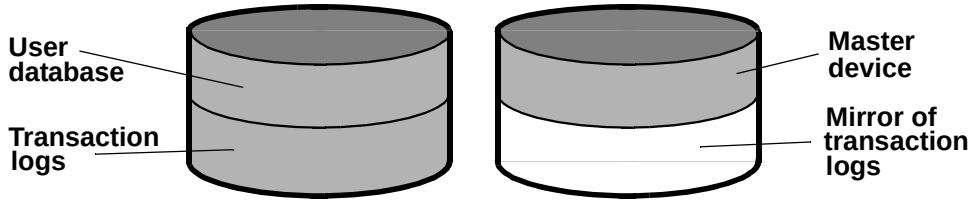
Mirroring using minimal physical disk space

Figure 2-1 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 2-1: Disk mirroring using minimal physical disk space



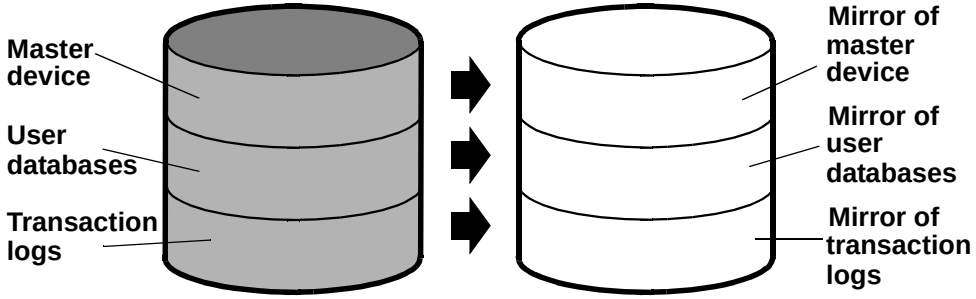
This configuration minimizes the amount of disk storage required. It provides for full recovery, even if the disk storing the user database and transaction log is damaged, because the mirror of the transaction log ensures full recovery. However, this configuration does not provide nonstop recovery because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

Mirroring for nonstop recovery

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

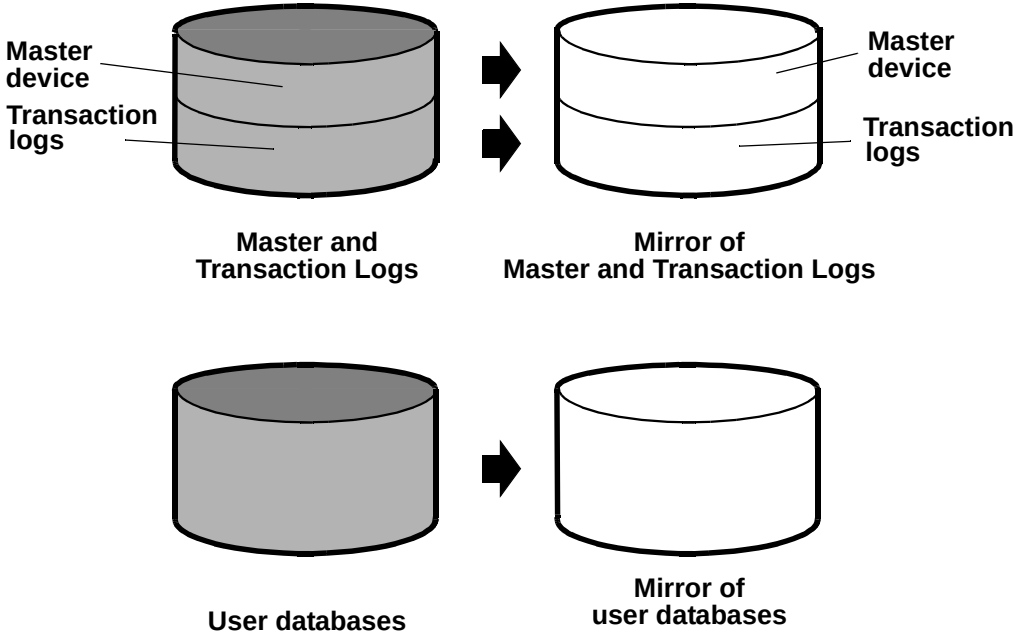
Figure 2-2: Disk mirroring for rapid recovery



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 2-3 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 2-3: 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 10, “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, dump the master database to ensure recovery in case master is damaged.

Initializing mirrors

disk mirror 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 works, but does not use asynchronous I/O.

The disk mirror syntax is:

```
disk mirror
    name = "device_name" ,
    mirror = "physicalname"
    [ , writes = { serial | noserial }]
```

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 creates 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.

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"
```

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 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 during hardware maintenance:

```
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 temporarily deactivates the specified device; 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 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 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 7-2 on page 254). Its effects on the phyname and mirrorname columns in sysdevices depend on the side argument also, as shown in Table 2-1.

Table 2-1: Effects of mode and side options to the disk mirror command

mode \ *side*	primary	secondary
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, you can include the waitfor mirrorexit command 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 environment, you must edit the runserver file for your Adaptive Server so that the mirror device starts when the server starts.

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

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

For information about mirroring the master device on Windows NT, see the *Utility Guide*.

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 sp_helpdevice.

Disk mirroring tutorial

This section illustrates the use of disk mirroring commands and their effect on selected columns of master..sysdevices. The status number and its hexidecimal equivalent for each entry in sysdevices are in parentheses:

Step 1 Initialize a new test device using:

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

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

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   NULL                   16386
```

Status 16386 indicates that the device is a physical device (2, 0x00000002), and any writes are to a UNIX file (16384, 0x00004000). Since the mirrorname column is null, mirroring is not enabled on this device.

Step 2 Mirror the test device using:

```
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  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

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  18658
```

Status 18658 indicates that the device is mirrored (64, 0x00000040), and the mirror device has been retained (2048, 0x00000800), but mirroring has been disabled (512 bit off), and only the primary device is used (256 bit off). Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file (16384, 0x00004000) and are in serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

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  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

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  16866
```

Status 16866 indicates that the device is mirrored (64, 0x00000040), but mirroring has been disabled (512 bit off) and that only the secondary device is used (256, 0x00000100). Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file (16384, 0x00004000), and are in serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

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  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

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.dat   NULL                     16386
```

Status 16386 indicates that the device is a physical device (2, 0x00000002), and any writes are to a UNIX file (16384, 0x00004000). 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.

Disk resizing and mirroring

You can use disk resize command only when mirroring is permanently disabled. If you try to run disk resize on a device that is mirrored, you see this:

```
disk resize can proceed only when mirroring is
permanently disabled. Unmirror secondary with mode =
'remove' and re-execute disk resize command.
```

When mirroring is only temporarily disabled, two scenarios can arise:

- The primary device is active while the secondary device is temporarily disabled and produces the same error as shown above.
- The secondary device is active while the primary device is temporarily disabled and produces an error with this message;

  ```
  disk resize can proceed only when mirroring is
  permanently disabled. Unmirror primary with mode
  = 'remove' and re-execute the command.
  ```

❖ **Increasing the size of a mirrored device**

1. Permanently disable mirroring on the device.
2. Increase the size of the primary device.
3. Physically remove the mirror device (in case of file).
4. Reestablish mirroring.

CHAPTER 3

Configuring Memory

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

Determining memory availability for Adaptive Server

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, 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 used by any client applications that run on the Adaptive Server machine. Windowing systems, such as X Windows, require a lot of memory and may interfere with Adaptive Server performance when used on the same machine as Adaptive Server.

The memory left over after subtracting requirements for the operating system and other applications is the total memory available for Adaptive Server. The value of the max memory configuration parameter specifies the maximum amount of memory to which Adaptive Server is configurable. See "Configuration parameters that affect memory allocation" on page 51 for information about configuring Adaptive Server to use this memory.

How Adaptive Server allocates memory

All database object pages are sized in terms of the **logical page size**, which you specify when you build a new master device. All databases—and all objects in every database—use the same logical page size. The size of Adaptive Server's logical pages (2, 4, 8, or 16K) determines the server's space allocation. Each allocation page, object allocation map (OAM) page, data page, index page, text page, and so on are built on a logical page. For example, if the logical page size of Adaptive Server is 8K, each of these page types are 8K in size. All of these pages consume the entire size specified by the size of the logical page. Larger logical pages allow you to create larger rows, which can improve your performance because Adaptive Server accesses more data each time it reads a page. For example, a 16K page can hold 8 times the amount of data as a 2K page, an 8K page holds 4 times as much data as a 2K page, and so on, for all the sizes for logical pages.

The logical page size is a server-wide setting; you cannot have databases with varying size logical pages within the same server. All tables are appropriately sized so that the row size is no greater than the current page size of the server. That is, rows cannot span multiple pages.

Regardless of the logical page size for which it is configured, Adaptive Server allocates space for objects (tables, indexes, text page chains) in extents, each of which is eight logical pages. That is, if a server is configured for 2K logical pages, it allocates one extent, 16K, for each of these objects; if a server is configured for 16K logical pages, it allocates one extent, 128K, for each of these objects.

This is also true for system tables. If your server has many small tables, space consumption can be quite large if the server uses larger logical pages. For example, for a server configured for 2K logical pages, systypes—with approximately 31 short rows, a clustered and a non-clustered index—reserves 3 extents, or 48K of memory. If you migrate the server to use 8K pages, the space reserved for systypes is still 3 extents, 192K of memory. For a server configured for 16K, systypes requires 384K of disk space. For small tables, the space unused in the last extent can become significant on servers using larger logical page sizes.

Databases are also affected by larger page sizes. Each database includes the system catalogs and their indexes. If you migrate from a smaller to larger logical page size, you must account for the amount of disk space each database requires. Table 3-1 lists the minimum size for a database on each of the logical page sizes.

Table 3-1: Minimum database sizes

Logical page size	Minimum database size
2K	2MB
4K	4MB
8K	8MB
16K	16MB

The logical page size is not the same as the memory allocation page size. Memory allocation page size is always 2K, regardless of logical page size, which can be 2, 4, 8, or 16K. Most memory-related configuration parameters use units of 2K for their memory page size. These configuration parameters include:

- max memory
- total logical memory

- total physical memory
- procedure cache size
- size of process object heap
- size of shared class heap
- size of global fixed heap

Disk space allocation

The logical page size is not the same as the memory allocation page size. This is the unit in which disk space is allocated, and Adaptive Server allocates this space in 2K pages. Some of the configuration parameters use this 2K page size for their allocation units.

Larger logical page sizes and buffers

Adaptive Server allocates buffer pools in units of logical pages. For example, on a server using 2K logical pages, 8MB are allocated to the default data cache. This constitutes approximately 2048 buffers. If you allocated the same 8MB for the default data cache on a server using a 16K logical page size, the default data cache is approximately 256 buffers. On a busy system, this small number of buffers might result in a buffer always being in the wash region, causing a slowdown for tasks requesting clean buffers. In general, to obtain the same buffer management characteristics on larger page sizes as with 2K logical page sizes, you should scale the size of the caches to the larger page size. So, if you increase your logical page size by four times, your cache and pool sizes should be about four times larger as well.

Adaptive Server typically allocates memory dynamically and allocates memory for row processing as it needs it, allocating the maximum size for these buffers, even if large buffers are unnecessary. These memory management requests may cause Adaptive Server to have a marginal loss in performance when handling wide-character data.

Heap memory

A heap memory pool is an internal memory pool created at start-up that tasks use to dynamically allocate memory as needed. This memory pool is used by tasks that requires a lot of memory from the stack, such as tasks that use wide columns. For example, if you make a wide column or row change, the temporary buffer this task uses can be as large as 16K, which is too big to allocate from the stack. Adaptive Server dynamically allocates and frees memory during the task's runtime. The heap memory pool dramatically reduces the predeclared stack size for each task, while also improving the efficiency of memory usage in the server. The heap memory the task uses is returned to the heap memory pool when the task is finished.

Set the heap memory with the heap memory per user configuration parameter. The syntax is:

```
sp_configure 'heap memory per user', amount_of_memory
```

Heap memory is measured in bytes per user. By default, the amount of memory is set to 4096 bytes. This example specifies setting the default amount of heap memory for 10 users:

```
sp_configure 'heap memory per user', 4096
```

You can also specify the amount of memory in the number of bytes per user. For example, the following example specifies that each user connection is allocated 4K bytes of heap memory:

```
sp_configure 'heap memory per user', 0, "4K"
```

At the initial Adaptive Server configuration, 1MB is set aside for heap memory. Additional heap memory is allocated for all the user connections and worker processes for which the server is configured, so the following configuration parameters affect the amount of heap memory available when the server starts:

- number of user connections
- number of worker processes

The global variable *@@heapmemsize* reports the size of the heap memory pool, in bytes.

Calculating heap memory

To calculate how much heap memory Adaptive Server sets aside, perform the following (Adaptive Server reserves a small amount of memory for internal structures, so these numbers vary from site to site):

((1024 * 1024) + (*heap memory in bytes*)* (number of user connections + number of worker processes))

The initial value of (1024 * 1024) is the 1MB initial size of the heap memory pool. Adaptive Server reserves a small amount of memory for internal structures.

For example, if your server is configured for:

- heap memory per user – 4K
- number of user connectins – 25 (the default)
- number of worker processes – 25 (the default)

@@heapmemsize reports 1378304 bytes.

And the estimated value using the formula above, is:

((1024 X 1024) + (4 * 1024 * 50)) = 1253376

Now, if you increase the number of user connections, the size of the heap memory pool increases accordingly:

```
sp_configure 'user connections', 100
```

@@heapmemsize reports 1716224 bytes.

The estimated value in this case comes out to be:

((1024 * 1024) + (4 * 1024 * (100 + 25)) = 1560576

If your applications were to fail with the following error message:

```
There is insufficient heap memory to allocate %ld
bytes. Please increase configuration parameter 'heap
memory per user' or try again when there is less
activity on the system.
```

You can increase the heap memory available to the server by increasing one of:

- heap memory per user
- number of user connections
- number of worker processes

Sybase recommends that you first try to increase the heap memory per user configuration option before you increase number of user connections or number of worker processes. Increasing the number of user connections and number of worker processes first consumes system memory for other resources, which may cause you to increase the server's maximum memory.

See Chapter 5, "Setting Configuration Parameters" for more information on how to make memory related configuration option changes.

The size of the memory pool depends on the number of user connections. Sybase recommends that you set heap memory per user to at least three times the size of your logical page.

How Adaptive Server uses memory

Memory exists in Adaptive Server as total logical or physical memory:

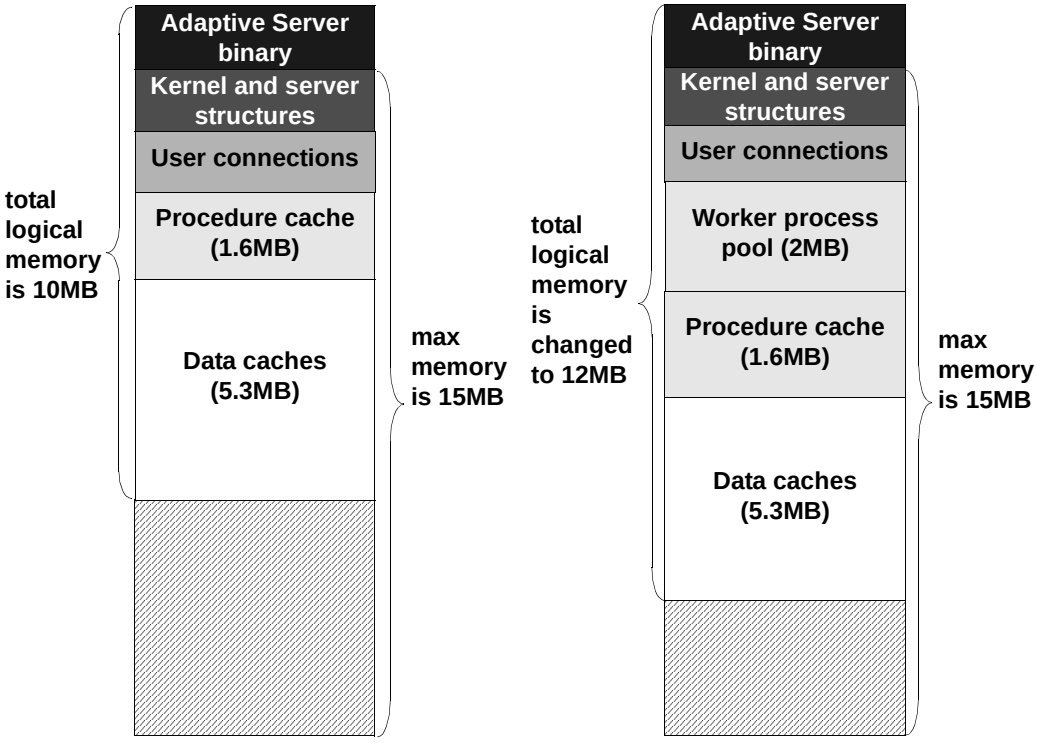
- Total logical memory – is the sum of the memory required for all the sp_configure parameters. The total logical memory is required to be available, but may or may not be in use at a given moment. The total logical memory value may change due to changes in the configuration parameter values.
- Total physical memory – is the sum of all shared memory segments in Adaptive Server. That is, total physical memory is the amount of memory Adaptive Server uses at a given moment. You can verify this value with the read-only configuration parameter total physical memory. The value of total physical memory can only increase because Adaptive Server does not shrink memory pools once they are allocated. You can decrease the amount of total physical memory by changing the configuration parameters and restarting Adaptive Server.

When Adaptive Server starts, it allocates memory for:

- Memory used by Adaptive Server for nonconfigurable data structures
- Memory for all user-configurable parameters, including the data cache, the procedure cache, and the default data cache.

Figure 3-1 illustrates how Adaptive Server allocates memory as you change some of the memory configuration parameters:

Figure 3-1: How Adaptive Server handles memory configuration changes



When a 2MB worker process pool is added to the Adaptive Server memory configuration, the procedure and data caches maintain their originally configured sizes; 1.6MB and 5.3MB, respectively. Because max memory is 5MB larger than the total logical memory size, it easily absorbs the added memory pool. If the new worker process pool brings the size of the server above the limit of max memory, any command you issue to increase the worker process pool fails. If this happens, the total logical memory required for the new configuration is indicated in the sp_configure failure message. Set the value of max memory to a value greater than the total logical memory required by the new configuration. Then retry your sp_configure request.

Note The values for max memory and total logical memory do not include the Adaptive Server binary.

The size of the default data cache and the procedure cache has a significant impact on overall performance. See Chapter 10, "Memory Use and Performance," in the *Performance and Tuning Guide: Basics* for recommendations on optimizing procedure cache size.

How much memory does Adaptive Server need?

The total memory Adaptive Server requires to start is the *sum of all memory configuration parameters* plus the *size of the procedure cache* plus the *size of the buffer cache,* where the size of the procedure cache and the size of the buffer cache are expressed in round numbers rather than in percentages. The procedure cache size and buffer cache size do *not* depend on the total memory you configure. You can configure the procedure cache size and buffer cache size independently. Use sp_cacheconfig to obtain information such as the total size of each cache, the number of pools for each cache, the size of each pool, and so on.

Use sp_configure to determine the total amount of memory Adaptive Server is using at a given moment. For example:

```
1> sp_configure "total logical memory"

Parameter Name        Default   Memory Used   Config Value   Run Value
   Unit               Type
--------------        --------------- --- ----------- ------------ ---------
--------------------- ----------
total logical memory  33792    127550         63775         63775
   memory pages(2k)   read-only
```

The value for the Memory Used column is represented in kilobytes, while the value for the Config Value column is represented in 2K pages.

The config value indicates the total logical memory Adaptive Server uses while it is running. The run value column shows the total logical memory being consumed by the current Adaptive Server configuration. You see a different output if you run this example because no two Adaptive Servers are likely to be configured in exactly the same fashion.

For more information about sp_configure, see the *Reference Manual Volume 3: Procedures.*

If you are upgrading

If you upgrade to version 12.5 Adaptive Server or higher, pre-12.5 Adaptive Server configuration values for total logical memory, procedure cache percent, and min online engines are used to calculate the new values for procedure cache size and number of engines at startup. Adaptive Server computes the size of the default data cache during the upgrade and writes this value to the configuration file. If the computed sizes of the data cache or procedure cache are less than the default sizes, they are reset to the default. During the upgrade, max memory is set to the value of total logical memory specified in the configuration file.

Reset the value of max memory to comply with the resource requirements.

You can use the verify option of sp_configure to verify any changes you make to the configuration file without having to restart Adaptive Server. The syntax is:

```
sp_configure "configuration file", 0, "verify", "full_path_to_file"
```

How much memory can Adaptive Server use?

The following table lists the upper limits of addressable shared memory for Adaptive Server versions 12.0.x and 12.5.x:

Table 3-2: Addressable memory limits by platform

Platform	32-bit Adaptive Server	64-bit Adaptive Server
HP-UX 11.x (PA-RISC processor)	2.75 giga bytes	16 exabytes[1]
IBM AIX 5.x	2.75 gigabytes	16 exabytes
Sun Solaris 8 (sparc processor)	3.78 gigabytes	16 exabytes
Sun Solaris 8 (Intel x86 processor)	3.75 gigabytes	N/A
Red Hat Enterprise Linux (Intel x86 processor)	2.7 gigabytes	N/A

[1]One exabyte equals 2^{60}, or 1024 PetaByte. 16 exabyte is a theoretical limit; in practice, the actual value is limited by the total memory available on the system. Adaptive Server has been tested with a maximum of 256 GB of shared memory.

[2]Starting Windows NT with the /3G option allows Adaptive Server to use up to 3 gigabytes of shared memory. For more information, see your Windows NT documentation.

Note The 12.5.x and later versions of Adaptive Server allocates memory differently than previous releases. This includes changes to existing memory-related configuration parameters and introduces new memory parameters. Review the new memory configuration parameters for Adaptive Server version 12.5.x before modifying the server or operating system memory parameters. For more information, see *What's New in Adaptive Server Enterprise?* and the *System Administration Guide* for details.

Each operating system has a default maximum shared-memory segment. Make sure the operating system is configured to allow the allocation of a shared-memory segment at least as large as Adaptive Server's max memory. For more information, see the Adaptive Server *Installation Guide* for your platform.

Configuration parameters that affect memory allocation

When setting the Adaptive Server memory configuration, you specify each memory requirement with an absolute value, using sp_configure. You also specify the size of the procedure and default data caches in an absolute value.

There are three configuration parameters that affect the way in which memory is allocated. They are: max memory, allocate shared memory, and dynamic allocation on demand.

max memory

max memory allows you to establish a maximum setting for the amount of memory you can allocate to Adaptive Server. Setting max memory to a slightly larger value than immediately necessary provides extra memory that is utilized when the Adaptive Server memory needs are increased.

The way Adaptive Server allocates the memory specified by max memory depends on how you configure allocate max shared memory and dynamic allocation on demand.

During upgrade, if the value for max memory is insufficient, Adaptive Server automatically increases the value for max memory. Upgrading to a newer version of Adaptive Server may require more memory because the size of internal data structures is increased.

allocate max shared memory

allocate max shared memory allows you to either allocate all the memory specified by max memory at start-up or to allocate only the memory required by the total logical memory specification during start-up.

On some platforms, if the number of shared memory segments allocated to an application is more than an optimal, platform-specific number, it could result in performance degradation. If this occurs, set max memory to the maximum amount available for Adaptive Server. Set allocate max shared memory to 1 and restart the server. This ensures that all the memory for max memory will be allocated by Adaptive Server during start time with the least number of segments.

For example, if you set allocate max shared memory to 0 (the default) and max memory to 500MB, but the server configuration only requires 100MB of memory at start-up, Adaptive Server allocates the remaining 400MB only when it requires the additional memory. However, if you set allocate max shared memory to 1, Adaptive Server allocates the entire 500MB when it starts.

The advantage of allocating all memory at start-up, by setting allocate max shared memory to 1, is that there is no performance degradation while the server is readjusting for the additional memory. However, if you do not predict memory growth properly, and max memory is set to a large value, you may be wasting physical memory. Since you cannot dynamically decrease memory configuration parameters, it is important that you take into account other memory requirements.

dynamic allocation on demand

dynamic allocation on demand allows you to determine whether your memory resources are allocated as soon as they are requested, or only as they are needed. Setting dynamic allocation on demand to 1 allocates memory changes as needed, and setting it to 0 allocates the configured memory requested in the memory configuration change at the time of the memory reconfiguration.

For example, if you set the value of dynamic allocation on demand to 1 and you change number of user connections to 1024, the total logical memory is 1024 multiplied by the amount of memory per user. If the amount of memory per user is 112K, then the memory for user connections is 112MB (1024 x 112).

This is the maximum amount of memory that the number of user connections configuration parameter is allowed to use. However, if only 500 users are connected to the server, the amount of total physical memory used by the number of user connections parameter is 56MB (500 x 112).

Now assume the value of dynamic allocation on demand is 0; when you change number of user connections to 1024, all user connection resources are configured immediately.

Optimally, organize Adaptive Server memory so that the amount of total physical memory is less than the amount of total logical memory, which is less than the max memory. This can be achieved, in part, by setting the value of dynamic allocation on demand to 1, and setting the value of allocate max shared memory to 0.

Dynamically allocating memory

Adaptive Server allocates physical memory dynamically. This allows users to change the memory configuration of Adaptive Server without restarting the server.

Note Adaptive Server allocates memory dynamically; however, it does not decrease memory dynamically. It is important that you accurately assess the needs of your system, because you may need to restart the server if you decrease the memory configuration parameters and want to release previously used physical memory. See "Dynamically decreasing memory configuration parameters" on page 54 for more information.

Consider changing the value of the max_memory configuration parameter when:

- You change the amount of RAM on your machine.
- The pattern of use of your machine changes.
- The configuration fails because max_memory is insufficient.

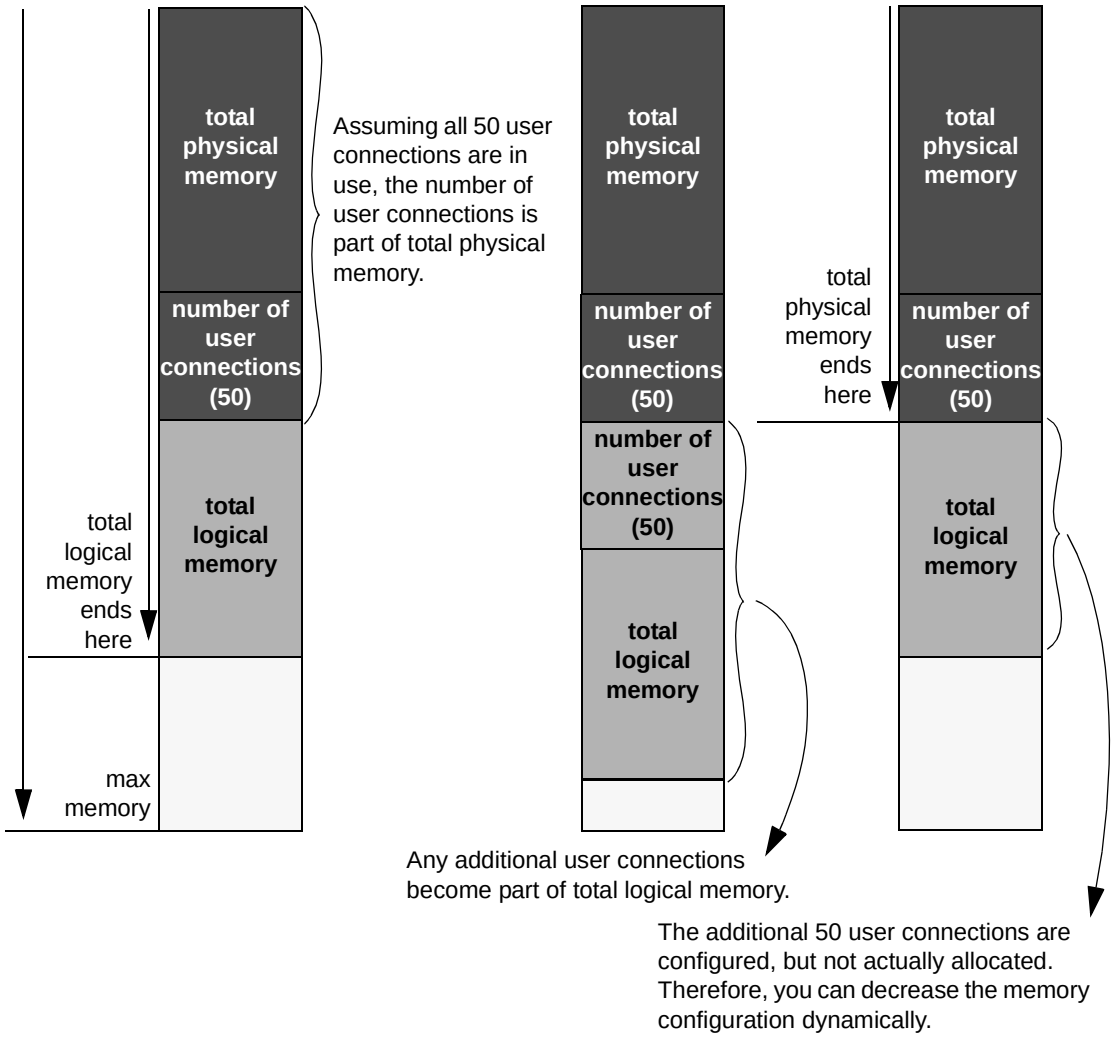
If Adaptive Server cannot start

When Adaptive Server starts, it must acquire the full amount of memory set by total logical memory. If Adaptive Server cannot start because it cannot acquire enough memory, reduce the memory requirements by reducing the values for the configuration parameters that consume memory. You may also need to reduce the values for other configuration parameters that require large amounts of memory. Restart Adaptive Server to use the new values. See Chapter 5, "Setting Configuration Parameters." for information about using configuration files.

Dynamically decreasing memory configuration parameters

If you reset memory configuration parameters to a lower value, any engaged memory is not released dynamically. To see how the changes in memory configuration are decreased, see Figure 3-2 and Figure 3-3.

Figure 3-2: dynamic allocation on demand set to 1 with no new user connections



In Figure 3-2, because dynamic allocation on demand is set to 1, memory is now used only when there is an event that triggers a need for additional memory use. In this example, such an event would be a request for additional user connections, when a client attempts to log in to Adaptive Server.

You may decrease number of user connections to a number that is greater than or equal to the number of user connections actually allocated, because, with dynamic allocation on demand set to 1, and without an actual increase in user connection request, no additional memory is required from the server.

Figure 3-3: dynamic allocation on demand set to 1, with new user connections logged on

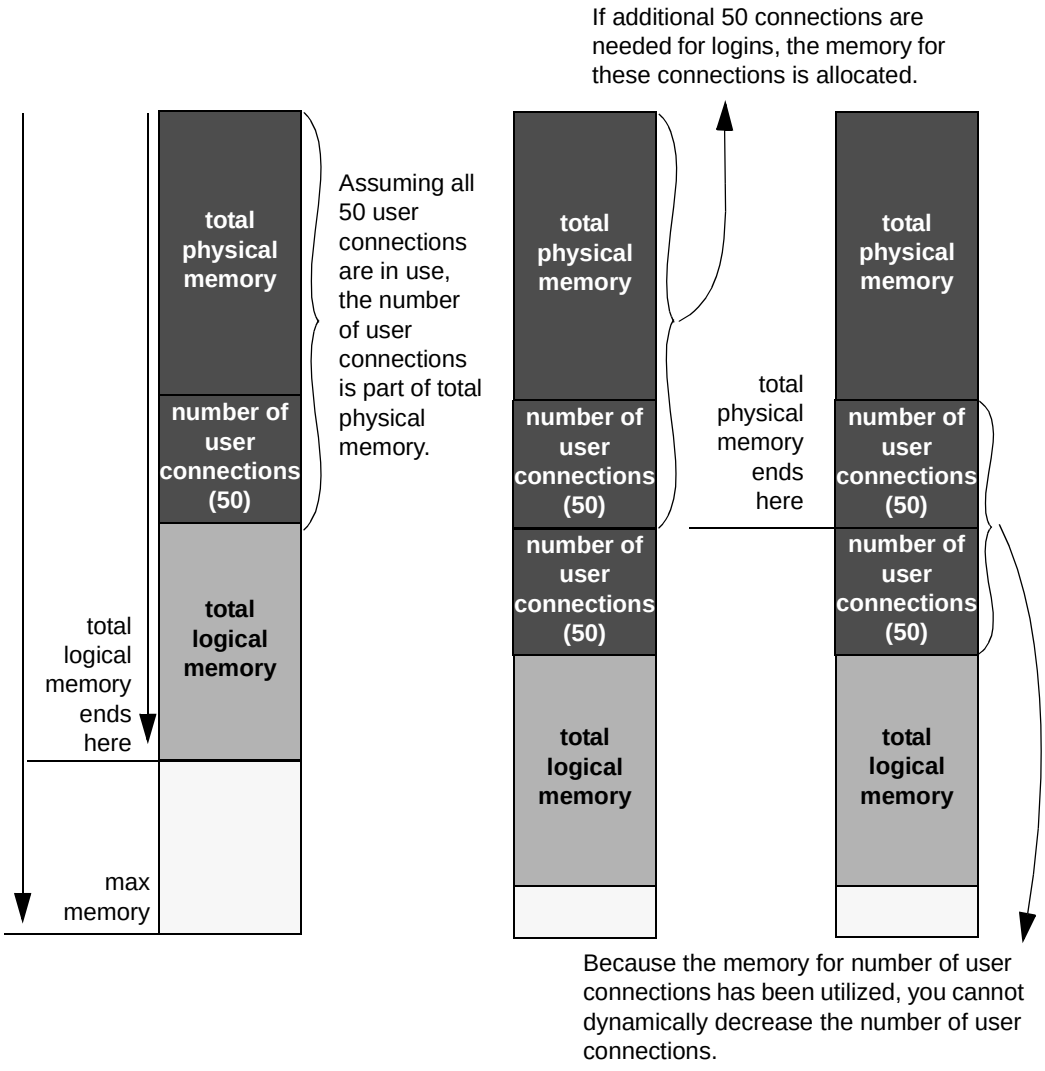
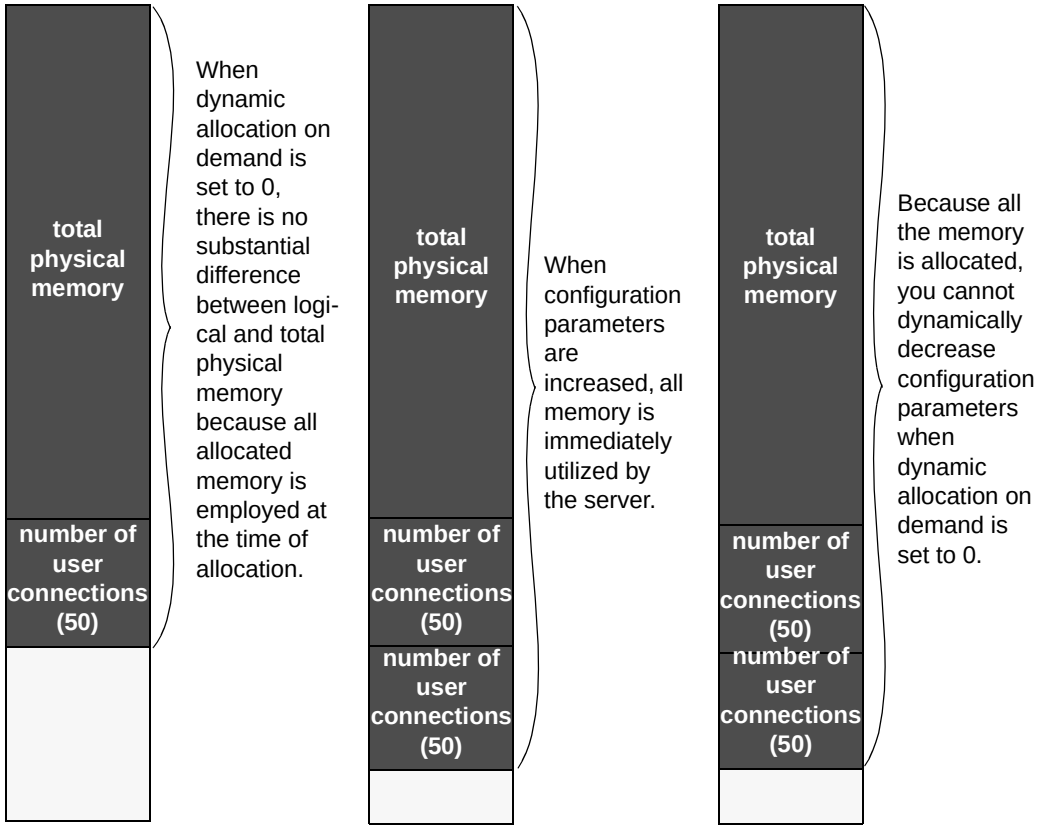


Figure 3-3 assumes that each of the additional 50 user connections is actually used. You cannot decrease number of user connections, because the memory is in use. You can use sp_configure to specify a change to memory configuration parameters, but this change does not take place until the server is restarted.

Figure 3-4: dynamic allocation on demand set to 0



Note In theory, when dynamic allocation on demand is set to 0, there should be no difference between total logical and physical memory. However, there are some discrepancies in the way that Adaptive Server estimates memory needs, and the way in which memory is actually required for usage. For this reason, you may see a difference between the two during runtime.

When dynamic allocation on demand is set to 0, all configured memory requirements are immediately allocated. You cannot dynamically decrease memory configuration.

In Figure 3-3 and Figure 3-4, users can change the memory configuration parameter values to any smaller, valid value. While this change does not take place dynamically, it disallows new memory use. For example, if you have configured number of user connections to allow for 100 user connections and then change that value to 50 user connections, in the situations represented by Figure 3-3 and Figure 3-4 , you can decrease the number of user connections value back to 50. This change does not Affect the memory used by Adaptive Server until after the server is restarted, but it prevents any new users from logging in to the server.

System procedures for configuring memory

The three system procedures to use for configuring Adaptive Server memory are:

- sp_configure
- sp_helpconfig
- sp_monitorconfig

The full syntax and usage of sp_configure and details on each configuration parameter are covered in see Chapter 5, “Setting Configuration Parameters.”

Using *sp_configure* to set configuration parameters

Execute sp_configure, specifying “Memory Use,” to see the parameters associated with memory use on Adaptive Server.

```
sp_configure "Memory Use"
```

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 and for number of worker processes, if this is more than 200.

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.

How much memory is available for dynamic growth?

Issuing sp_configure memory displays all of the memory parameters and determines the difference between max memory and total logical memory, which is the amount of memory available for dynamic growth. For example:

```
1> sp_configure memory

Msg 17411, Level 16, State 1:
Procedure 'sp_configure', Line 187:
Configuration option is not unique.
Parameter Name                 Default  Memory Used  Config Value  Run Value
Unit                 Type
------------------------------ ----------- ----------- ------------ ----------
--------------------- ----------
additional network memory            0            0             0           0
bytes                dynamic
allocate max shared memory           0            0             0           0
switch               dynamic
heap memory per user              4096            0          4096        4096
bytes                dynamic
lock shared memory                   0            0             0           0
switch               static
max memory                       33792       300000        150000      150000
memory pages(2k)     dynamic
memory alignment boundary        16384            0         16384       16384
bytes                static
memory per worker process         1024            4          1024        1024
bytes                dynamic
shared memory starting address       0            0             0           0
not applicable       static
total logical memory             33792       110994         55497       55497
memory pages(2k)     read-only
```

```
total physical memory                0        97656            0       48828
memory pages(2k)    read-only

An additional 189006 K bytes of memory is available for reconfiguration. This is
the difference between 'max memory' and 'total logical memory'.
```

Using *sp_helpconfig*

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:

```
1> 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
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
0              2147483647     0              0              0
number   dynamic

Configuration parameter, 'number of worker processes', will consume 7091K of
memory if configured at 50.
Changing the value of 'number of worker processes' to '50' increases the amount
of memory ASE uses by 7178 K.
```

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:

```
1> 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
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
5              2147483647     25             25             3773
number  dynamic
Configuration parameter, 'number of user connections', can be configured to 33
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 cannot calculate memory use. sp_helpconfig prints an error message in these cases. For example, if you attempt to specify a size for a parameter that toggles, such as allow resource limits, sp_helpconfig prints the message that describes the function of the parameter for all the configuration parameters that do not use memory.

Using *sp_monitorconfig*

sp_monitorconfig displays metadata cache usage statistics on certain shared server resources, including:

- 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
- The current size of the procedure cache and the amount actually used

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.

```
1> sp_monitorconfig "number of open indexes"

Usage information at date and time: Apr 22 2002  2:49PM.
Name             num_free  num_active  pct_act  Max_Used  Reused
--------------   --------  ----------  -------  --------  ------
number of open        217         283    56.60       300  No
```

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.

You can also determine the current size of the procedure cache with sp_monitorconfig procedure cache size. This parameter describes the amount of space in the procedure cache is currently configured for and the most it has ever actually used. For example, the procedure cache in the following server is configured for 20,000 pages:

```
1> sp_configure "procedure cache size"

option_name                    config_value run_value
------------------------------ ------------ ---------
procedure cache size                   3271      3271
```

However, when you run sp_montorconfig “procedure cache size”, you find that the most the procedure cache has ever used is 14241 pages, which means that you can lower the run value of the procedure cache, saving memory:

```
1> sp_monitorconfig "procedure cache size"

Usage information at date and time: Apr 22 2002  2:49PM.
Name               num_free  num_active  pct_act  Max_Used  Reused
--------------     --------  ----------  -------  --------  ------
procedure cache         5878       14122    70.61     14241  No
```

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. These parameters should be checked by System Administrators who are configuring an Adaptive Server for the first time. System Administrators should review these parameters when the system configuration changes, after upgrading to a new version 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 68.

Adaptive Server executable code size

The size of the executable code is not included in the value of total logical memory or max memory calculations. total logical memory reports the actual memory requirement for Adaptive Server configuration, excluding any memory required for the executable.

The memory required for the executable is managed by the operating system and is beyond the control of Adaptive Server. Consult your operating system documentation for more details.

Data and procedure caches

As explained in "How Adaptive Server uses memory" on page 47, you specify the size of the data and procedure caches. Having sufficient data and procedure cache space is one of the most significant contributors to performance. This section explains the details between the two caches and how to monitor cache sizes.

Determining the procedure cache size

procedure cache size specifies the size of your procedure cache in 2K pages, regardless of the server's logical page size. For example:

```
1> sp_configure "procedure cache size"
Parameter Name        Default  Memory Used  Config Value  Run Value
```

```
Unit              Type
-------------------  -------  -----------  ------------  ---------
---------------  ---------
procedure cache size     3271         6914          3271      3271
memory pages(2k)  dynamic
```

The amount of memory used for the procedure cache is 8.248MB. To set the procedure cache to a different size, issue the following:

```
sp_configure "procedure cache size", new_size
```

This example resets the procedure cache size to 10000 2K pages (20MB):

```
sp_configure "procedure cache size", 10000
```

Determining the default data cache size

Both sp_cacheconfig and sp_helpcache display the current default data cache in megabytes. For example, the following shows an Adaptive Server configured with 19.86MB of default data cache:

```
sp_cacheconfig


Cache Name               Status     Type       Config Value    Run Value
------------------       --------   --------   ------------    ----------
default data cache       Active     Default    0.00 Mb         19.86Mb
                                                ------------    --------
                        Total                   0.00Mb          19.86 Mb
===========================================================================
Cache: default data cache, Status: Active, Type: Default
       Config Size: 0.00 Mb, Run Size: 19.86 Mb
       Config Replacement: strict LRU, Run Replacement: strict LRU
       Config Partition:    1,   Run Partition:    1
IO Size       Wash Size     Config Size     Run Size      APF Percent
--------      ---------     ------------    ----------    -----------
2 Kb          4066 Kb        0.00 Mb         19.86 Mb               10
```

To change the default data cache, issue sp_cacheconfig, and specify "default data cache." For example, to change the default data cache to 25MB, enter:

```
sp_cacheconfig "default data cache", "25M"
```

This change is dymamic.

The default data cache size is an absolute value, and the minimum size for the cache size is 256 times the logical page size; for 2K, the minimum is 512K: for 16K, the minimum is 4M. The default value is 8MB. During the upgrade process, Adaptive Server sets the default data cache size to the value of the default data cache in the configuration file.

Monitoring cache space

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

```
sp_configure "total data cache size"
```

Monitoring cache sizes using the error log

Another way to determine how Adaptive Server uses memory is to examine the memory-related messages written to the 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 allocation on Adaptive Server. As discussed earlier, the amount of memory allocated for the procedure caches depends on the run value of the procedure cache size 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.

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.

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
```

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

Changing any of these parameters changes the amount of space used by each user connection: you 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.

Open databases, open indexes, and open objects

The configuration parameters that control the total number of databases, indexes, partitions and objects that can be open at one time are managed by the **metadata caches**. The metadata caches reside in the server structures portion of Adaptive Server's memory. You configure space for each of these caches with these parameters (For information about these parameters, see Setting Configuration Parameters in the System Administration Guide, Volume 1):

- number of open databases
- number of open indexes
- number of open objects
- number of open partitions

When Adaptive Server opens a database or accesses an index, partition, or object, it reads information about it in the corresponding system tables: The information for databases in in sysdatabases, the information for indexes in in sysindexes, the information for partitions is in syspartitions and so on.

The metadata caches for databases, indexes, partitions or objects allow Adaptive Server to access the information that describe them in sysdatabases, sysindexes, syspartitions or sysobjects 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, partition, or object information at runtime.

Managing individual metadata caches for databases, indexes, partitions, or objects is beneficial for a database that contains a large number of indexes, partitions, and objects and where there is high concurrency among users.

Number of locks

All processes in Adaptive Server share a pool of lock structures. As a first estimate for configuring the number of locks, 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 171 for more information. For information on how worker processes use memory, see "Worker processes" on page 69.

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 165 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 106 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 or worker processes
- memory per worker processes
- 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 these parameters changes the amount of memory required for each worker process:

- defaule network packet size
- stack size and stack guard size
- user log cache size
- memory per worker process

memory per worker process 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 the *Performance and Tuning Guide: Basics* 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.

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

Number of remote sites

Set number of remote sites to the number of simultaneous sites you must 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

These 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	• max online engines
• allow resource limits	• max SQL text monitored

• audit queue size	• number of alarms
• event buffers per engine	• number of large i/o buffers
• max number network listeners	• permission cache entries

The statement cache

The statement cache is used for saving SQL of cached statements. Adaptive Server compares incoming SQL to its cached SQL, and if they are equal, it executes the plan of the SQL already saved. This allows the application to amortize the costs of query compilation across several executions of the same statement.

Setting the statement cache

The statement cache allows Adaptive Server to store the text of ad hoc SQL statements. Adaptive Server compares a newly received ad hoc SQL statement to cached SQL statements and, if a match is found, uses the plan cached from the initial execution. In this way, Adaptive Server does not have to recompile SQL statements for which it already has a plan.

The statement cache is a server-wide resource, which allocates and consumes memory from the procedure cache memory pool. Set the size of the statement cache dynamically using the statement cache size configuration parameter.

Note If you deallocate or reduce the amount of memory for the statement cache, the original memory allocated is not released until you restart Adaptive Server.

The syntax is as follows, where *size_of_cache* is the size, in 2K pages:

```
sp_configure "statement cache size", size_of_cache
```

For example, to set your statement cache to 5000 2K pages, enter:

```
sp_configure "statement cache size", 5000
```

See Chapter 5, “Setting Configuration Parameters,” for more information.

Consider the following when you configure memory for the statement cache:

- The amount of memory allocated for the procedure cache memory pool is the sum of the statement cache size and the procedure cache size configuration parameters. The statement cache memory is taken from the procedure cache memory pool. In the example above, the size of the procedure cache memory pool is increased by 5000 2K pages.
- statement cache size limits the amount of procedure cache memory available for cached SQL text and plans. That is, Adaptive Server cannot use more memory for the statement cache than you have configured with the statement cache size configuration parameter.
- All procedure cache memory, including that memory allocated by the statement cache size configuration parameter, is available for stored procedures, which may replace cached statements on an LRU basis.
- Increase the max memory configuration parameter by the same amount configured for the statement cache. That is, if you have initially configured the statement cache size to be 100 2K pages, increase max memory by the same amount.
- If you have configured the statement cache with the statement cache size configuration parameter, you can disable and enable the statement cache at the session level with set statement cache. By default, the statement cache is on at the session level if it has been configured at the server level.
- Because each cached statement consumes one object descriptor, you must also increase the number of object descriptors accordingly, using the number of open databases configuration parameter. To estimate how many cached SQL statements to allow for, see "Statement cache sizing" on page 74.

Ad hoc query processing

Adaptive Server performs the following steps to process ad hoc SQL statements using the statement cache:

1 Adaptive Server parses the statement.

If the statement should be cached (see "Caching conditions" on page 74), Adaptive Server computes a hash value from the statement. Adaptive Server uses this hash value to search for a matching statement in the statement cache (see "Statement matching criteria" on page 73).

- If a match is found in the statement cache, Adaptive Server skips to step 4.
- If a match is not found, Adaptive Server proceeds to step 2.

2 Adaptive Server caches the SQL statement text.

3 Adaptive Server wraps the SQL statement with a lightweight stored procedure and changes any local variables into procedure parameters. The internal representation of the lightweight procedure is not yet compiled into the plan.

4 Adaptive Server converts the SQL statement into an execute statement for the corresponding lightweight procedure.

- If there is no plan in the cache, Adaptive Server compiles the procedure and caches the plan. Adaptive Server compiles the plan using the assigned runtime values for the local variables.
- If the plan exists but is invalid, Adaptive Server returns to step 3 using the text of the cached SQL statement.

5 Adaptive Server then executes the procedure.

Statement matching criteria

Adaptive Server matches an ad hoc SQL statement to a cached statement by the SQL text and by login (particularly if both users have sa_role), user ID, database ID, and session state settings. The relevant session state consists of settings for the following set command parameters:

- forceplan
- jtc
- parallel_degree
- prefetch
- quoted_identifier
- sort_merge
- table count

- transaction isolation level
- chained (transaction mode)

Settings for these parameters determine the behavior of the plan Adaptive Server produces for a cached statement. For information on the set command and its parameters, see the *Adaptive Server Reference Manual*.

Note You must configure set chained on/off in its own batch if you enable the statement cache.

Caching conditions

- Adaptive Server currently caches select, update, delete, and insert select statements with at least one table reference.
- Statements are not cached if the abstract plan dump or abstract plan load parameters are enabled.
- Adaptive Server does not cache select into statements, cursor statements, dynamic statements, plain insert (not insert select) statements, and statements within stored procedures, views, and triggers. Statements that refer to temporary tables are not cached, nor are statements with language parameters transmitted as BLOB datatypes. Statements that are prohibitively large are not cached. Also, select statements that are part of a conditional if exists or if not exists clause are not cached.

Statement cache sizing

Each cached statement requires approximately 1K memory in the statement cache, depending on the length of the SQL text. Each cached plan requires at least 2K of memory in the procedure cache. To estimate the statement cache memory required, account for the following for each statement to be cached:

- The length of the SQL statement, in bytes, rounded up to the nearest multiple of 256.
- Approximately 100 bytes overhead.

- The size of the plan in the procedure cache. This size is equivalent to the size of a stored procedure plan containing only the cached statement. There may be duplicates of the plan for a single cached statement being used concurrently by two or more users.

Monitoring the statement cache

sp_sysmon reports on statement caching and stored procedure executions. The statement cache is monitored by the following counters:

- Statements Found in Cache – the number of times a query plan was reused. A low number of cache hits may indicate the statement cache is too small.
- Statements Not Found – indicates a lack of repeated SQL statements. The sum of statements found in cache and statements not found is the total number of eligible SQL statements submitted.
- Statements Cached – the number of SQL statements added to the cache. This is typically the same number as Statements Not Found. Smaller values for statements cached means the statement cache is full of active statements.
- Statements Dropped – the number of statements that were dropped from the cache. A high value may indicate an insufficient amount of procedure cache memory or that you have configured a statement cache size that is too small.
- Statements Restored – the number of query plans regenerated from the SQL text. High values indicate an insufficient procedure cache size.
- Statements Not Cached – the number of statements Adaptive Server would have cached if the statement cache were enabled. However, Statements Not Cached does not indicate how many unique SQL statements would be cached.

This is sample output from sp_sysmon:

```
Procedure Cache Management        per sec     per xact       count  % of total
--------------------------  -----------  ------------  ----------  ----------
  Procedure Requests                6.6           3.7          33         n/a
  Procedure Reads from Disk         1.0           0.6           5       15.2%
  Procedure Writes to Disk          0.4           0.2           2        6.1%
  Procedure Removals                2.6           1.4          13         n/a
  Procedure Recompilations          0.8           0.4           4         n/a
```

```
Recompilations Requests:
 Execution Phase                0.6             0.3             3         75.0%
 Compilation Phase              0.2             0.1             1         25.0%
 Execute Cursor Execution       0.0             0.0             0          0.0%
 Redefinition Phase             0.0             0.0             0          0.0%

Recompilations Reasons:
 Table Missing                  0.6             0.3             3          n/a
 Temporary Table Missing        0.2             0.1             1          n/a
 Schema Change                  0.0             0.0             0          n/a
 Index Change                   0.0             0.0             0          n/a
 Isolation Level Change         0.2             0.1             1          n/a
 Permissions Change             0.0             0.0             0          n/a
 Cursor Permissions Change      0.0             0.0             0          n/a

SQLStatement Cache:
 Statements Cached              0.0             0.0             0          n/a
 Statements Found in Cache      0.7             0.0             2          n/a
 Statements Not Found           0.0             0.0             0          n/a
 Statements Dropped             0.0             0.0             0          n/a
 Statements Recompiled          0.3             0.0             1          n/a
 Statements Not Cached          1.3             0.0             4          n/a
```

Purging the statement cache

Run dbcc purgesqlcache to remove all the SQL statements from the statement cache. Any statements that are currently running are not removed.

You must have the sa_role to run dbcc purgesqlcache

dbcc purgesqlcache prints the following message:

```
dbcc purgesqlcache
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator
(SA) role.
```

Printing statement summaries

Run dbcc prsqlcache to print summaries of the statements in the statement cache. The *oid* option allows you to specify the object ID of the statement to print, and the *printopt* option allows you to specify whether you print the trace description (specify 0) or the showplan option (specify 1). If you do not include any values for *oid* or *printopt*, dbcc prsqlcache displays the entire contents of the statement cache.

You must have the sa_role to run dbcc prsqlcache

This provides information for all statements in the cache:

```
dbcc prsqlcache
Start of SSQL Hash Table at 0xfc67d830
Memory configured: 1000 2k pages             Memory used: 18 2k pages
Bucket# 625 address 0xfc67ebb8

SSQL_DESC 0xfc67f9c0
ssql_name *ss1248998166_0290284638ss*
ssql_hashkey 0x114d645e ssql_id 1248998166
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 8
SQL TEXT: select * from sysobjects where name like "sp%"

Bucket# 852 address 0xfc67f2d0
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

End of SSQL Hash Table

DBCC execution completed. If DBCC printed error messages, contact a user with
```

Or you can get information about a specific object ID:

```
dbcc prsqlcache (1232998109, 0)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
```

```
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

This example specifies 1 in the printopt parameter for the showplan output:

```
dbcc prsqlcache (1232998109, 1)

SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1       ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

QUERY PLAN FOR STATEMENT 1 (at line 1).

      STEP 1
         The type of query is SELECT.
         FROM TABLE
            systypes
         Nested iteration.
         Table Scan.
         Forward scan.

         Positioning at start of table.
         Using I/O Size 2 Kbytes for data pages.
         With LRU Buffer Replacement Strategy for data pages.
         DBCC execution completed. If DBCC printed error messages,
         contact a user with
          System Administrator (SA) role.
```

Configuring memory for caches

Memory is the most important consideration when you are configuring Adaptive Server. Memory is consumed by various configuration parameters, the procedure cache, statement cache, and data caches. Setting the values of the various configuration parameters and the caches correctly is critical for good system performance.

The total memory allocated during system start-up is the sum of memory required for all the configuration needs of Adaptive Server. This value can be obtained from the read-only configuration parameter total logical memory. This value is calculated by Adaptive Server. The configuration parameter max memory must be greater than or equal to total logical memory. max memory indicates the amount of memory you will allow for Adaptive Server needs.

During server start-up, by default, Adaptive Server allocates memory based on the value of total logical memory. However, if the configuration parameter allocate max shared memory has been set, then the memory allocated will be based on the value of max memory. The configuration parameter allocate max shared memory enables a System Administrator to allocate the maximum memory that is allowed to be used by Adaptive Server, during server start-up.

The key points for memory configuration are:

- The System Administrator should determine the size of shared memory available to Adaptive Server and set max memory to this value.
- The configuration parameter allocate max shared memory can be turned on during start-up and runtime to allocate all the shared memory up to max memory with the least number of shared memory segments. A large number of shared memory segments has the disadvantage of some performance degradation on certain platforms. Check your operating system documentation to determine the optimal number of shared memory segments. Once a shared memory segment is allocated, it cannot be released until the server is restarted.
- The difference between max memory and total logical memory is additional memory available for the procedure and statement caches, data caches, or for other configuration parameters.

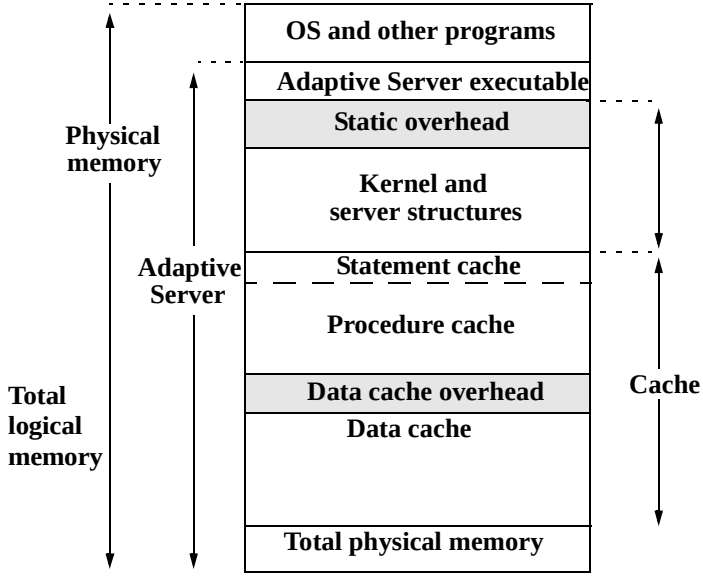
  The amount of memory to be allocated by Adaptive Server during start-up, is determined by either total logical memory or max memory. If this value too high:

  - Adaptive Server may not start if the physical resources on your machine are not sufficient.
  - If it does start, the operating system page fault rates may rise significantly and the operating system may need to be reconfigured to compensate.

- Handling wider character literals requires Adaptive Server to allocate memory for string user data. Also, rather than statically allocating buffers of the maximum possible size, Adaptive Server allocates memory dynamically. That is, it allocates memory for local buffers as it needs it, always allocating the maximum size for these buffers, even if large buffers are unnecessary. These memory management requests may cause Adaptive Server to have a marginal loss in performance when handling wide-character data.
- If you require Adaptive Server to handle more than 1000 columns from a single table, or process over 10000 arguments to stored procedures, the server must set up and allocate memory for various internal data structures for these objects. An increase in the number of small tasks that are performed repeatedly may cause performance degradation for queries that deal with larger numbers of such items. This performance hit increases as the number of columns and stored procedure arguments increases.
- Memory that is allocated dynamically slightly degrades the server's performance.
- When Adaptive Server uses larger logical page sizes, all disk I/Os are performed in terms of the larger logical page sizes. For example, if Adaptive Server uses an 8K logical page size, it retrieves data from the disk in 8K blocks. This should result in an increased I/O throughput, although the amount of throughput is eventually limited by the controller's I/O bandwidth.

What remains after all other memory needs have been met is available for the procedure and statement cache, and the data cache. Figure 3-5 shows how memory is divided.

Figure 3-5: How ASE Replicator uses memory



CHAPTER 4

Configuring Data Caches

This chapter describes how to create and administer named caches on Adaptive Server.

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 10, “Memory Use and Performance,” in the *Performance and Tuning Guide: Basics* discusses performance concepts associated with data caches.

The Adaptive Server data cache

The data cache holds the data, index, and log pages currently in use, as well as pages used recently by Adaptive Server. When you install Adaptive Server, it has a single default data cache that is used for all data, index, and log activity. The default size of this cache is 8M. Creating other caches does not reduce the size of the default data cache. 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 128K on a server configured with 16K logical pages means that Adaptive Server can read an entire extent—8 pages—all at once, rather than performing 8 separate I/Os.

Sorts can also take advantage of buffer pools configured for large I/O sizes.

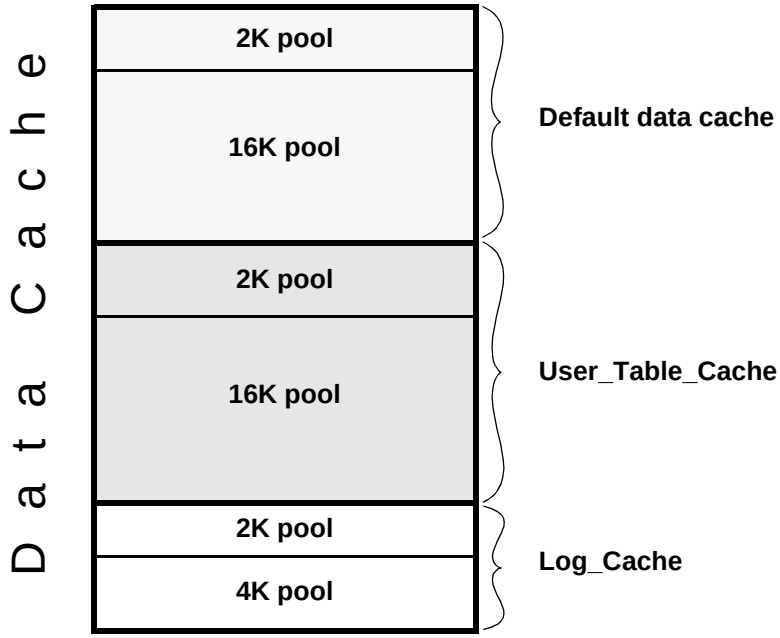
Configuring named data caches does not divide the default cache into separate cache structures. The named data caches that you create can be used only by databases or database objects that are explicitly bound to them. All objects not explicitly bound to named data caches use the default data cache.

Adaptive Server provides user-configurable data caches to improve performance, especially for multiprocessor servers. For information about how the data cache affects performance, see “Data cache” of the *Performance and Tuning Guide: Basics*.

Figure 4-1 shows a cache with the default data cache and two named data caches. This server uses 2K logical pages.

The default cache contains a 2K pool and a 16K pool. The User_Table_Cache cache also has a 2K pool and a 16K pool. The Log_Cache has a 2K pool and a 4K pool.

Figure 4-1: Data cache with default cache and two named data caches



Cache configuration commands

Table 4-2 lists commands for configuring named data caches, for binding and unbinding objects to caches, and for reporting on cache bindings. It also lists procedures you can use to check the size of your database objects, and commands that control cache usage at the object, command, or session level.

Table 4-1: Commands for configuring named data caches

Command	Function
sp_cacheconfig	Creates or drops named caches, and changes the size, cache type, cache policy, or number of cache partitions.
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.

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 reenables 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 the cache to which a table is bound.
sp_helpindex	Reports the cache to which an index is bound.
sp_helpdb	Reports the cache to which a database is bound.
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.

Most of the parameters for sp_cacheconfig that configure the data cache are dynamic and do not require that you restart the server for them to take effect. See Table 4-2 on page 89 for a description of static and dynamic actions.

In addition to using the commands to configure named data caches interactively, you can also edit the configuration file located in the *$SYBASE* directory. However, this requires that you restart the server. See "Configuring data caches with the configuration file" on page 117 for more information.

Information on data caches

Use sp_cacheconfig to create and configure 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
```

The results of sp_cacheconfig look similar to:

```
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
     Config Partition:            1,   Run Partition:            1

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. For each cache, there is a block of information reporting the configuration for the memory pools in the cache.

The columns are:

- 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 but not yet active.
  - "Active" – the cache is currently active.
  - "Pend/Del" – the cache is being deleted. The cache size was reset to 0 interactively. See "Configuring data caches" on page 88 for more information.
- 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 currently configured value. 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.

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 is 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. The default size of the pool is the size of the server's logical page. When you first configure a cache, all the space is assigned to the 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 106 for more information.
- Config Size and Run Size – display the configured size and the size currently in use. These may differ for other pools if you have tried to move space between them, and some of the space could not be freed.
- Config Partition and Run Partition – display the configured number of cache partitions and the number of partitions currently in use. These may differ if you have changed the number of partitions since last restart.
- 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

The default data cache and the procedure cache for Adaptive Server are specified using an absolute value. The first step in planning cache configuration and implementing caches is to set the max memory configuration parameter. After you set max memory, determine how much space to allocate for data caches on your server. The size of a data cache is limited only by access to memory on the system; however, max memory should be larger than total logical memory. You must specify an absolute value for the size of the default data cache and all other user-defined caches. For an overview of Adaptive Server memory usage, see Chapter 3, "Configuring Memory."

You can configure data caches in two ways:

- Interactively, using sp_cacheconfig and sp_poolconfig. This method is dynamic and does not require a restart of Adaptive Server.
- By editing your configuration file. This method is static and requires that you restart Adaptive Server.

Each time you change the data cache or execute either 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.

Some of the actions you perform with sp_cacheconfig are dynamic and some are static.

Table 4-2: Dynamic and static sp_cacheconfig actions

Dynamic sp_cacheconfig actions	**Static sp_cacheconfig actions**
Adding a new cache	Changing the number of cache partitions
Adding memory to an existing cache	Reducing a cache size
Deleting a cache	Changing replacement policy
Changing a cache type	

You can reset static parameters by deleting and re-creating the cache:

1 Unbind the cache.

2 Delete the cache.

3 Re-create the cache using the new configuration.

4 Bind objects to the cache.

The following sections describe how to use sp_cacheconfig and sp_poolconfig. See "Configuring data caches with the configuration file" on page 117 for information about using the configuration file.

Mixing static and dynamic parameters

You can specify both static and dynamic parameters in a single command. Adaptive Server performs the dynamic changes immediately and writes all changes to the configuration file (both static and dynamic). Static changes take effect the next time the server is started.

Creating a new cache

The syntax for sp_cacheconfig is:

```
sp_cacheconfig [cachename [ ,"cache_size[P|K|M|G]" ]
      [,logonly | mixed ] [,strict | relaxed ] ]
      [, "cache_partition=[1|2|4|8|16|32|64]"]
```

Size units can be specified with:

- P – pages (Adaptive Server logical page size)
- K – kilobytes (default)
- M – megabytes
- G – gigabytes

See the *Adaptive Server Reference Manual* for a full description of the sp_cacheconfig syntax.

Maximum data cache size is limited only by the amount of memory available on your system. The memory required to create the new cache is taken from the Adaptive Server global memory. When the cache is created:

- It has a default wash size.
- The asynchronous prefetch size is set to the value of global async prefetch limit.
- It has only the default buffer pool.

Use sp_poolconfig to reset these values.

To create 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. The cache is immediately active. Running sp_cacheconfig now yields:

```
sp_cacheconfig
Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
default data cache             Active    Default       0.00 Mb      8.00 Mb
pubs_cache                     Active    Mixed        10.00 Mb     10.00 Mb
------------ ------------
Total      10.00 Mb     18.00 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
Config Size: 0.00 Mb,   Run Size: 8.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   1636 Kb      0.00 Mb      8.00 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
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

The pubs_cache is now active, and all space is assigned to the smallest pool.

Note When you create a new cache, the additional memory you specify is validated against max memory. If the sum of total logical memory and additional memory requested is greater than max memory, then Adaptive Server issues an error and does not perform the changes.

You can create as many caches as you want without restarting Adaptive Server.

Insufficient space for new cache

If Adaptive Server cannot allocate all the memory requested, it allocates all the available memory and issues the following message:

```
ASE is unable to get all the memory requested (%d). (%d) kilobytes have been
allocated dynamically.
```

However, this additional memory is not allocated until the next restart of Adaptive Server.

Adaptive Server notifies you of insufficient space is because some of the memory is unavailable because of resource constraints. System Administrators should make sure these resource constraints are temporary. If the behavior persists, a subsequent restart may fail.

For example, if max memory is 700MB, pub_cache is 100MB, and the server's total logical memory is 600MB, and you attempt to add 100MB to pub_cache, the additional memory fits into max memory. However, if the server can allocate only 90MB, then it allocates this amount dynamically, but the `size field` of the cache in the configuration file is updated to 100MB. On a subsequent restart, since Adaptive Server obtains memory for all data caches at one time, the size of pub_cache is 100MB.

Adding memory to an existing named cache

The syntax to add memory to an existing cache is:

sp_cacheconfig *cache_name*, "*new_size*[P|K|M|G]"

The syntax is described in "Creating a new cache" on page 89.

The additional memory you allocate is added to the Adaptive Server page size pool. For example, in a server with a logical page size of 4K, the smallest size for a pool is a 4K buffer pool. If the cache has partitions, the additional memory is divided equally among them.

If there is insufficient memory available, Adaptive Server allocates what it can and then allocates the full amount at the next reboot. See "Insufficient space for new cache" on page 91 for more information.

The following adds 2MB to a cache named pub_cache (the current size is 10MB):

```
sp_cacheconfig pub_cache, "12M"
```

sp_cacheconfig displays the following after the memory is added:

```
sp_cacheconfig pub_cache
 Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
pub_cache                      Active    Mixed      12.00 Mb       12.00
------------ ------------
Total      12.00 Mb     12.00 Mb
================================================================================
Cache: pub_cache,   Status: Active,   Type: Mixed
Config Size: 12.00 Mb,   Run Size: 12.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2456 Kb      0.00 Mb     12.00 Mb     10
```

This change adds memory to the database page-size buffer pool and recalculates the wash size, as required. If the absolute value is set for wash size, Adaptive Server does not recalculate it.

Decreasing the size of a cache

Note When you reduce the size of a cache, you must restart Adaptive Server for the change to take effect.

The 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
     Config Partition:            1,   Run Partition:            1

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 to 6MB from a current size of 7MB:

```
sp_cacheconfig pubs_log, "6M"
```

After a restart of Adaptive Server, sp_cacheconfig shows:

```
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
     Config Partition:            1,   Run Partition:            1

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 default pool (which is the smallest size available). You may need to move space to the default pool from other pools before you can reduce the size of the data cache. In the last example, to reduce the size of the cache to 3MB, use sp_poolconfig to move some memory into the default pool of 2K from the 4K pool. The memory is moved to "memory available for named caches." See "Changing the size of memory pools" on page 111 for more information.

Deleting a cache

To completely remove a data cache, reset its size to 0:

```
sp_cacheconfig pubs_log, "0"
```

The cache is immediately deleted.

Note You cannot drop the default data cache.

If you delete a data cache, and there are objects bound to the cache, the cache is left in memory and Adaptive Server issues the following message:

```
Cache cache_name not deleted dynamically. Objects are bound to the cache. Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

The entry corresponding to the cache in the configuration file is deleted, as well as the entries corresponding to the cache in sysconfigures, and the cache is deleted the next time Adaptive Server is restarted.

Use sp_helpcache to view all items bound to the cache. Use sp_unbindcache_all to unbind objects. See the *Adaptive Server Reference Manual: Procedures* for more information.

If you re-create the cache and restart Adaptive Server, the bindings are marked valid again.

Explicitly configuring the default cache

You must explicitly configure the size of the default data cache because it is specified with an absolute value. Use sp_helpcache to see the amount of memory remaining that can be used for the cache. For example:

```
sp_helpcache
```

```
Cache Name                 Config Size    Run Size        Overhead
-----------------------    ------------  ----------      ----------
default data cache         50.00 Mb       50.00 Mb        3.65 Mb
pubs_cache                 10.00 Mb       10.00 Mb        0.77 Mb

Memory Available For       Memory Configured
Named Caches               To Named Caches
--------------------       ----------------
91.26 Mb                   91.25 Mb

------------------ Cache Binding Information: ------------------

Cache Name       Entity Name    Type          Index Name     Status
----------         -----------      -----           ---------        ------
```

To specify the absolute size of the default data cache, execute sp_cacheconfig with default data cache and a size value. This command sets the default data cache size to 25MB:

```
sp_cacheconfig "default data cache", "25M"
```

When you rerun sp_helpconfig, “Config Value” shows the value.

```
sp_cacheconfig
Cache Name                 Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache         Active    Default      25.00 Mb     50.00 Mb
pubs_cache                 Active    Mixed        10.00 Mb     10.00 Mb
                                                 ------------ ------------
                                       Total      10.00 Mb     60.00 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
    Config Size: 25.00 Mb,   Run Size: 50.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb 10110 Kb     00.00 Mb    50.00 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
    Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
```

```
    2 Kb    2048 Kb     0.00 Mb     10.00 Mb     10
```

You can change the size of any named cache using sp_cacheconfig. Any changes you make to the size of any data cache do not affect the size of any other cache. Similarly, once you specify the size of the default data cache, the configuration of other user-defined caches does not alter the size of the default data cache.

Note If you configure the default data cache and then reduce max memory to a level that sets the total logical memory value higher than the max memory value, Adaptive Server does not start. Edit your configuration file to increase the size of other caches and increase the values of configuration parameters that require memory to create an environment in which total logical memory is higher than max memory. See Chapter 3, "Configuring Memory."for more information.

The default data cache and all user-defined caches are explicitly configured with an absolute value. In addition, many configuration parameters use memory. To maximize performance and avoid errors, set the value of max memory to a level high enough to accommodate all caches and all configuration parameters that use memory.

Adaptive Server issues a warning message if you set max memory to a value less than total logical memory.

Changing the cache type

To reserve a cache for use by only the transaction log, change the cache's type to "logonly." The change is dynamic.

This example creates the cache pubs_log with the type "logonly:"

```
sp_cacheconfig pubs_log, "7M", "logonly"
```

This shows the initial state of the cache:

```
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
```

In high-transaction environments, Adaptive Server usually performs best if the default value is twice the server's logical page size (for a server with 2K logical page size, it is 4K, for a server with 4K logical page size, it is 8K, and so on).. For larger page sizes (4, 8, and 16K) use sp_sysmon to find the optimum configuration for your site. For information on configuring caches for improved log performance, see "Matching log I/O size for log caches" on page 101.

Configuring cache replacement policy

If a cache is dedicated to a table or an index, and the cache has little or no buffer replacement when the system reaches a stable state, you can set the relaxed LRU (least recently used) replacement policy. The relaxed LRU replacement policy can improve performance for caches where there is little or no buffer replacement occurring, and for most log caches. See "Data cache" in the *Performance and Tuning Guide: Basics* for more information.

To set relaxed replacement policy, use:

```
sp_cacheconfig pubs_log, relaxed
```

The default value is "strict."

Note Setting the cache replacement policy is not dynamic and requires a restart of Adaptive Server to take effect.

You can create a cache and specify its cache type and the replacement policy in one command. These examples create two caches, pubs_log and pubs_cache:

```
sp_cacheconfig pubs_log, "3M", logonly, relaxed
sp_cacheconfig pubs_cache, "10M", mixed, strict
```

The change takes place immediately and is made to every cache partition of the named cache.

Here are the results:

```
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
     Config Partition:            1,   Run Partition:            1

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
     Config Partition:            1,   Run Partition:            1

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
     Config Partition:            1,   Run Partition:            1

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. The minimum size of a memory pool is the size of the server's logical page. Memory pools larger than this must be a power of two and can be a maximum size of one extent.

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 memory pool. Creating additional pools reassigns some of that space to other pools, reducing the size of the default memory 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.

After 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.

The commands that move space between pools within a cache do not require a restart of Adaptive Server, 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 sets 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 (this is the smallest size available). 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"
```

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
     Config Partition:             1,   Run Partition:             1
```

```
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
     Config Partition:              1,   Run Partition:              1

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 that is 8MB:

```
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
     Config Partition:            1,   Run Partition:            1

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 need not 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 twice the server's logical page size (for a server with 2K logical page size, it is 4K, for a server with 4K logical page size, it is 8K, and so on). Adaptive Server uses 2K I/O for the log if a 4K pool is not available. You can change the log I/O size with 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 "Choosing the I/O size for the transaction log" in the *Performance and Tuning Guide: Basics* for information on tuning the log I/O size.

Binding objects to caches

sp_bindcache assigns a database, table, index, text object or 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 the master database.
- 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.

Binding objects to caches is dynamic.

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 sp_dboption from master, and a use database command, and run checkpoint:

```
sp_dboption pubs2, single, true
```

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.
- A cursor is open on the object

In addition, Adaptive Server must 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 116 for more information.

Getting information about cache bindings

sp_helpcache 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            10.00 Mb       10.00 Mb        0.77 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.

Checking cache overhead

Note The cache overhead accounting for Adaptive Server versions 12.5.1 and later is more explicit than earlier versions of Adaptive Server. The actual overhead is the same, but instead of being part of the server overhead, it is now considered as part of the 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 global memory block pool.

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 example checks the overhead for 20,000 pages:

```
sp_helpcache "20000P"

2.96Mb of overhead memory will be needed to manage a
cache of size 20000P
```

You are not wasting any cache space by configuring user caches. Approximately 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 86 shows a default data cache with 59.44 MB of cache space available before any user-defined caches are created. The server in this example uses a 2K logical page. When the 10MB pubs_cache is created, the results of sp_cacheconfig show a total cache size of 59.44 MB.

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"

4.10Mb 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"

0.73Mb of overhead memory will be needed to manage a
cache of size 10M
```

This command shows the overhead for a cache size of 49.44MB:

```
sp_helpcache "49.44M"
3.46Mb of overhead memory will be needed to manage a
cache of size 49.44M
```

4.19MB (which is equal to .73M + 3.46M) is the required overhead size for maintaing two caches of size 10MB and 49.44MB, and is slightly more than the 4.10M overhead that necessary to maintain one cache of 59.44MB.

Cache sizes are rounded to two places when printed by sp_cacheconfig, and overhead is rounded to two places by sp_helpcache, so you 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 commands unbinds the titleidind index on the titles table in the pubs2 database:

```
sp_unbindcache pubs2, titles, titleidind
```

To unbind all the objects bound to a cache, use sp_unbindcache_all, giving the cache's name:

```
sp_unbindcache_all pubs_cache
```

Note You cannot use sp_unbindcache_all if more than eight databases 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

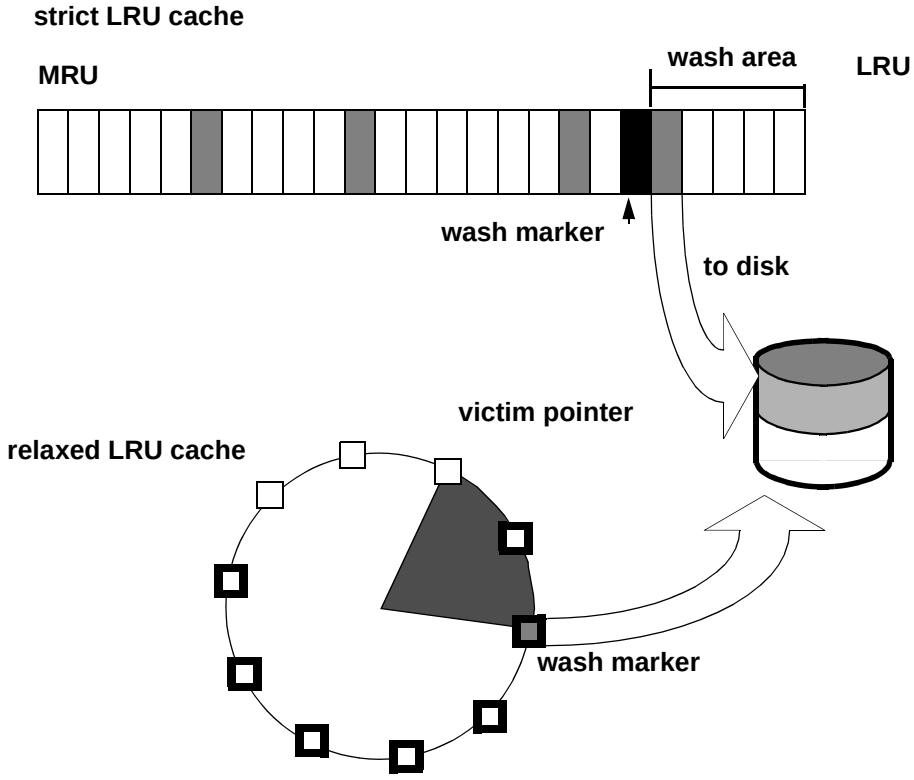
When Adaptive Server needs to read a buffer into cache, it places the buffer:

- At the LRU (least recently used) end of each memory pool, in a cache with strict LRU policy.
- 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**. After 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 4-2 illustrates how the wash area of a buffer pool works with a strict and relaxed LRU cache.

Figure 4-2: 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 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.

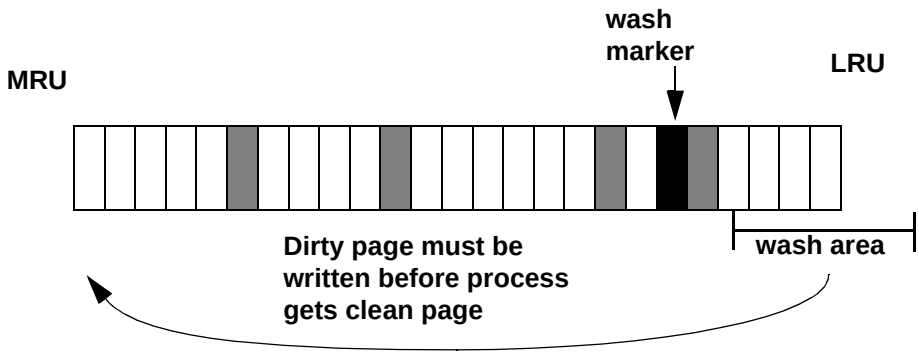
A buffer is a block of pages that matches the I/O size for the pool. Each buffer is treated as a unit: all pages in the buffer are read into cache, written to disk, and aged in the cache as a unit. For the size of the block, multiply the number of buffers by the pool size—for a 2K pool, 256 buffers equals 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 to 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 impact performance. Figure 4-3 shows a dirty buffer grab on a strict replacement policy cache.

Figure 4-3: Small wash area results in a dirty buffer grab



You can use 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 8K memory pool to 720K:

```
sp_poolconfig pubs_cache, "8K", "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.

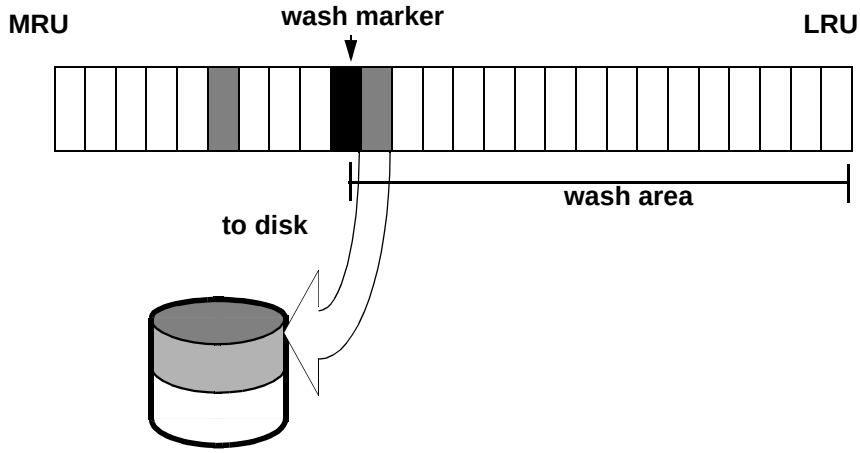
See the *Performance and Tuning Guide: Monitoring* for more information.

When the wash area is too large

If the wash area in a pool 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 4-4. 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 the *Performance and Tuning Guide: Monitoring* for more information.

Figure 4-4: Effects of making the wash area too large



Setting housekeeper to avoid washes for cache

You can specify that you do not want the housekeeper to perform a wash on a particular cache with the HK ignore cache option of the cache status parameter in the configuration file. Specifying which caches to not include in the wash allows you to avoid contention between the housekeeper and cache manager spinlock. You manually set HK ignore cache in the configuration file under each cache's heading; you cannot set HK ignore cache with sp_cacheconfig.

For example, the following show the settings for the named cache newcache:

```
Named Cache:newcache
     cache size = 5M
     cache status = mixed cache
     cache status = HK ignore cache
     cache replacement policy = DEFAULT
local cache partition number = DEFAULT
```

You must set HK ignore cache along with either of default data cache, mixed cache, or log parameters. You cannot set the parameter cache status to only HK ignore cache. For example, the following is not allowed:

```
Named Cache:newcache
     cache size = 5M
     cache status = HK ignore cache
     cache replacement policy = DEFAULT
local cache partition number = DEFAULT
```

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 global async prefetch limit configuration parameter. 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 10, “Tuning Asynchronous Prefetch,” in *Performance and Tuning Guide: Optimizer and Abstract Plans*.

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 pool.

Moving space from the memory pool

This command checks the current configuration of the pubs_log cache (the output in this example is based on the examples in the previous sections):

```
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
     Config Partition:            1,   Run Partition:            1

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
     Config Partition:            1,   Run Partition:            1
```

```
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 another pool 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
     Config Partition:            1,   Run Partition:            1

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 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
```

```
     Config Partition:            1,   Run Partition:            1

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.

Adding cache partitions

On multi-engine servers, more than one task can attempt to access the cache at the same time. By default, each cache has a single spinlock, so that only one task can change or access the cache at a time. If cache spinlock contention is above 10 percent, increasing the number of cache partitions for a cache can reduce spinlock contention, and improve performance.

You can configure the number of cache partitions for:

- All data caches, using the global cache partition number configuration parameter
- An individual cache, using sp_cacheconfig

The number of partitions in a cache is always a power of 2 between 1 and 64. No pool in any cache partition can be smaller than 512K. In most cases, since caches can be sized to meet requirements for storing individual objects, you should use the local setting for the particular cache where spinlock contention is an issue.

See "Reducing spinlock contention with cache partitions" in the *Performance and Tuning Guide: Basics* for information on choosing the number of partitions for a cache.

Setting the number of cache partitions with *sp_configure*

Use sp_configure to set the number of cache partitions for all caches on a server. For example, to set the number of cache partitions to 2, enter:

```
sp_configure "global cache partition number",2
```

You must restart the server for the change to take effect.

Setting the number of local cache partitions

Use sp_cacheconfig or the configuration file to set the number of local cache partitions. This command sets the number of cache partitions in the default data cache to 4:

```
sp_cacheconfig "default data cache", "cache_partition=4"
```

You must restart the server for the change to take effect.

Precedence

The local cache partition setting always takes precedence over the global cache partition value.

These commands set the server-wide partition number to 4, and the number of partitions for pubs_cache to 2:

```
sp_configure "global cache partition number", 4
sp_cacheconfig "pubs_cache", "cache_partition=2"
```

The local cache partition number takes precedence over the global cache partition number, so pubs_cache uses 2 partitions. All other configured caches have 4 partitions.

To remove the local setting for pubs_cache, and use the global value instead, use this command:

```
sp_cacheconfig "pubs_cache", "cache_partition=default"
```

To reset the global cache partition number to the default, use:

```
sp_configure "global cache partition number", 0, "default"
```

Dropping a memory pool

To completely remove a pool, reset its size to 0. The following removes the 16K pool and places all space in the default pool:

```
sp_poolconfig "default data cache", "0", "16K"
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
     Config Partition:            1,   Run Partition:            1

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 default pool. You cannot delete the default pool in any cache.

When pools cannot be dropped due to pages use

If the pool you are trying to delete contains pages that are in use, or pages that have dirty reads, but are 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 one 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 removing 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

To bind or unbind user tables, indexes, or text or image objects, the cache binding commands must have an exclusive table lock on the object. If a user holds locks on a table, and you issue 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. 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 this 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 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 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 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 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.

This 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      29.28 Mb     25.00 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      59.28 Mb     55.00 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 29.28 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

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
     Config Partition:            1,   Run Partition:            1

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
     Config Partition:            1,   Run Partition:            1
```

```
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
     Config Partition:            1,   Run Partition:            1

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 = 29.28M
         cache status = default data cache
        cache replacement policy = DEFAULT
        local cache partition number = 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
        local cache partition number = DEFAULT

[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
        local cache partition number = DEFAULT

[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
        local cache partition number = DEFAULT
```

For more information about the configuration file, see Chapter 5, "Setting Configuration Parameters."

Warning! Check the max memory configuration parameter and allow enough memory for other Adaptive Server needs. If you assign too much memory to data caches in your configuration file, Adaptive Server does not start. If this occurs, edit the configuration file to reduce the amount of space in the data caches, or increase the max memory allocated to Adaptive Server. see Chapter 5, "Setting Configuration Parameters," for suggestions on monitoring cache sizes.

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 10, "Memory Use and Performance," in the *Performance and Tuning Guide: Basics*.

Here are some general guidelines:

- The size of the default data cache does not decrease when you configure other caches.

- 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 default cache is active. Transactions logs are read 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: 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 sp_logiosize. The default value is 4K.
- Trying to 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. 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.

Configuration file errors

If you edit your configuration file manually, 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 max 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.
```

CHAPTER 5

Managing Multiprocessor Servers

This chapter provides guidelines for administering Adaptive Server on a multiprocessor.

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. You can run Adaptive Server 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 20, "Managing Multiprocessor Servers," in the *Performance and Tuning Guide: Basics*.

Definitions

Here are the definitions for 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** – 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** – 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.

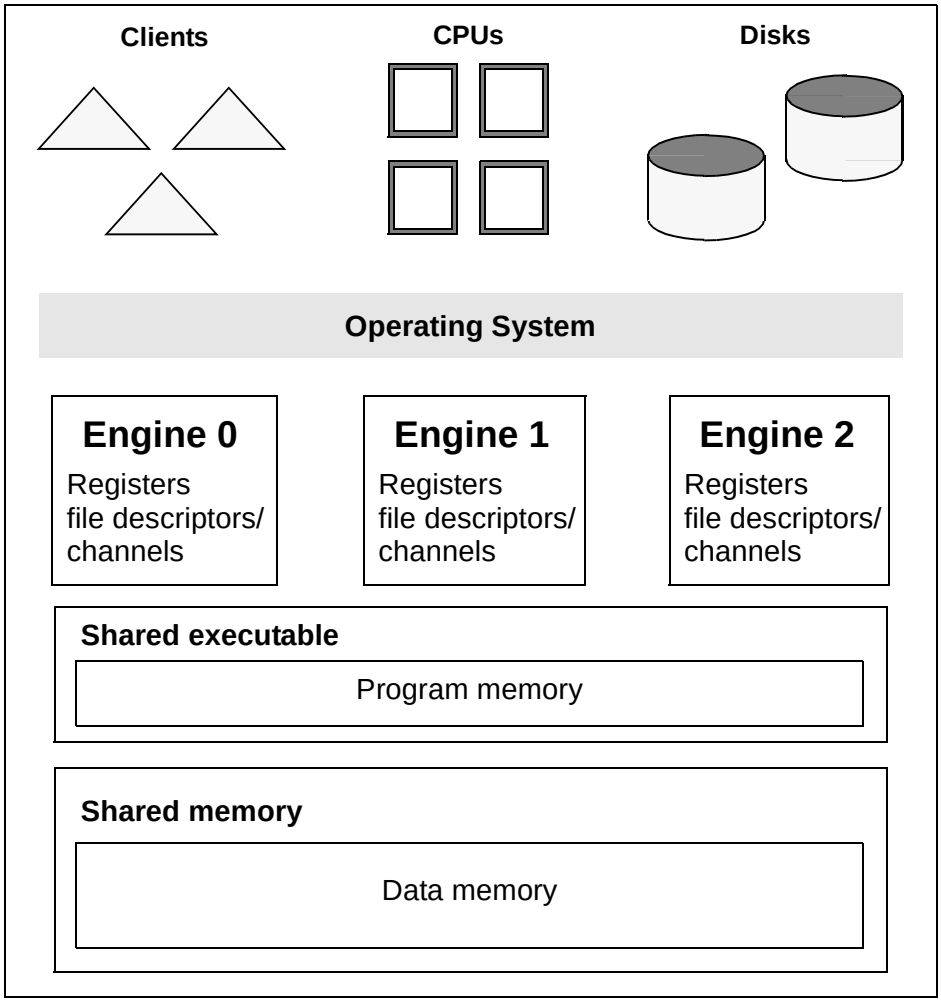
Target architecture

The SMP environment product is intended for machines with the following features:

- A symmetric multiprocessing operating system
- Shared memory over a common bus
- 1–128 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 5-1.

Figure 5-1: SMP environment architecture



When clients connect to Adaptive Server, the client connections are assigned to engines in a round-robin fashion, so all engines share the work of handling network I/O for clients. 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.

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 any Adaptive Server engine can pick up any Adaptive Server task, unless it is deliberately configured otherwise.

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 correct number of engines.

An engine represents a certain amount of CPU power. It is a configurable resource like memory.

Note If your server connections use Component Integration Services, they are affinitied to a single engine, and are not allowed to migrate from one engine to another. Adaptive Server uses a load-balancing algorithm to evenly distribute the load among the engines.

Resetting the number of engines

When you first install Adaptive Server, the system is configured for a single engine. To use multiple engines, reset the number of engines the first time you restart 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.

- Decrease the number of engines if Adaptive Server is not fully utilizing the existing engines. There is overhead involved in running the engines, and Adaptive Server is wasting resources if you don't need the extra engines.

max online engines controls the number of engines used by Adaptive Server. Reset this parameter with sp_configure. For example, to set the number of engines to 3, issue the following:

```
sp_configure "max online engines", 3
```

You must restart the server to reset the number of engines.

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 that you choose the correct number of engines for Adaptive Server. Here are some guidelines:

- Adaptive Server will not let you configure more engines than CPUs 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 is exceeded and bottlenecks may result.

Starting and stopping engines

This section describes how to start and stop Adaptive Server engines using sp_engine.

Monitoring engine status

Before you bring an engine online or offline, you may need to check the status of the engines currently running. sysengines includes any of the following in the status column:

- online – indicates the engine is online.
- in offline – indicates that sp_engine offline has been run. The engine is still allocated to the server, but is in the process of having its tasks migrated to other engines.
- in destroy – indicates that all tasks have successfully migrated off the engine, and that the server is waiting on the OS level task to deallocate the engine.
- in create – indicates that an engine is in the process of being brought online.
- dormant – indicates that sp_engine offline was not able to migrate all tasks from that engine. If the tasks terminate themselves (through a timeout), the engines switch to being permanently offline. dormant engines only process those tasks that are causing the dormant state; they are not available to work on any other tasks.

The following command shows the engine number, status, number of tasks affinitied, and the time an engine was brought online:

```
   select engine, status, affinitied, starttime
   from sysengines
engine status       affinitied  starttime
------ ------------ ----------- --------------------------
     0 online                12        Mar  5 2001 9:40PM
     1 online                 9        Mar  5 2001 9:41PM
     2 online                12        Mar  5 2001 9:41PM
     3 online                14        Mar  5 2001 9:51PM
     4 online                 8        Mar  5 2001 9:51PM
     5 in offline            10        Mar  5 2001 9:51PM
```

Starting and stopping engines with *sp_engine*

You can dynamically stop or start engines using sp_engine, which allows a System Administrator to reconfigure CPU resources as processing requirements fluctuate over time.

The syntax for sp_engine is:

sp_engine {“online” | [offline | can_offline] [, *engine_id*] | [“shutdown”, *engine_id*]

For example, the following brings engine 1 online. Messages are platform specific (in this example, Sun Solaris was used):

```
sp_engine "online", 1
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device connection
limit is 3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security modules loaded
successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 1, os pid 8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel asynchronous
disk I/O strategy
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck: Network fc0330c8 online
```

You can check whether or not a specific engine can be brought offline with the can_offline parameter. The following determines whether engine 1 can be brought offline:

```
sp_engine can_offline, 1
```

sp_engine specifies a return code of 0 if you can bring the specified engine offline. If you do not specify an *engine_id*, sp_engine describes the status of the engine in sysengines with the highest *engine_id*.

Engines can be brought online only if max online engines is greater then the current number of engines with an online status, and if enough CPU is available to support the additional engine.

To bring an engine offline, enter the engine ID. The following takes engine number one offline:

```
sp_engine offline, 1
```

Adaptive Server waits for any tasks that are associated with this engine to finish before taking the engine offline, and returns a message similar to the following:

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting for affinitied
process(es) before going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid 21127  offline
```

You cannot take engine number zero offline.

sp_engine "shutdown" forces any tasks associated with the specified engine to finish in a five-second period, and then shuts down the engine. You can use sp_engine shutdown when an engine has gone into a dormant state. Although it is perfectly acceptable to leave an engine in dormant state, you can use this command if you need to bring an engine offline. sp_engine kills any remaining processes that are preventing the engine from going offline normally. The following shuts down engine 1:

```
sp_engine "shutdown", 1
```

For more information about sp_engine, see the *Reference Manual*.

Relationship between network connections and engines

Due to the operating system limit on the number of file descriptors per process on UNIX, reducing the number of engines reduces the number of network connections that the server can have. On Windows NT, the number of network connections is independent of the number of engines.

There is no way to migrate a network connection created for server-to-server remote procedure calls—or example, connections to Replication Server and XP Server—so you cannot take an engine offline that is managing one of these connections.

Logical process management and *dbcc engine(offline)*

If you are using logical process management to bind particular logins or applications to engine groups, use dbcc engine(offline) carefully. If you take all engines for an engine group offline:

- The login or application can run on any engine
- An advisory message is sent to the connection logging in to the server

Since engine affinity is assigned when a client logs in, users who are already logged in are not migrated if the engines in the engine group are brought online again with dbcc engine("online").

Managing user connections

This section describes how to manage user connections in the UNIX environment.

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 these 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

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 the number of user connections parameter 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 5, "Setting Configuration Parameters."

Configuration parameters that affect SMP systems

Chapter 5, "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 5-1 lists system resources protected by spinlocks and the configuration parameters you can use to change the default spinlock ratio.

Table 5-1: Spinlock ratio configuration parameters

Configuration parameter	System resource protected
lock spinlock ratio	Number of lock hash buckets
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:

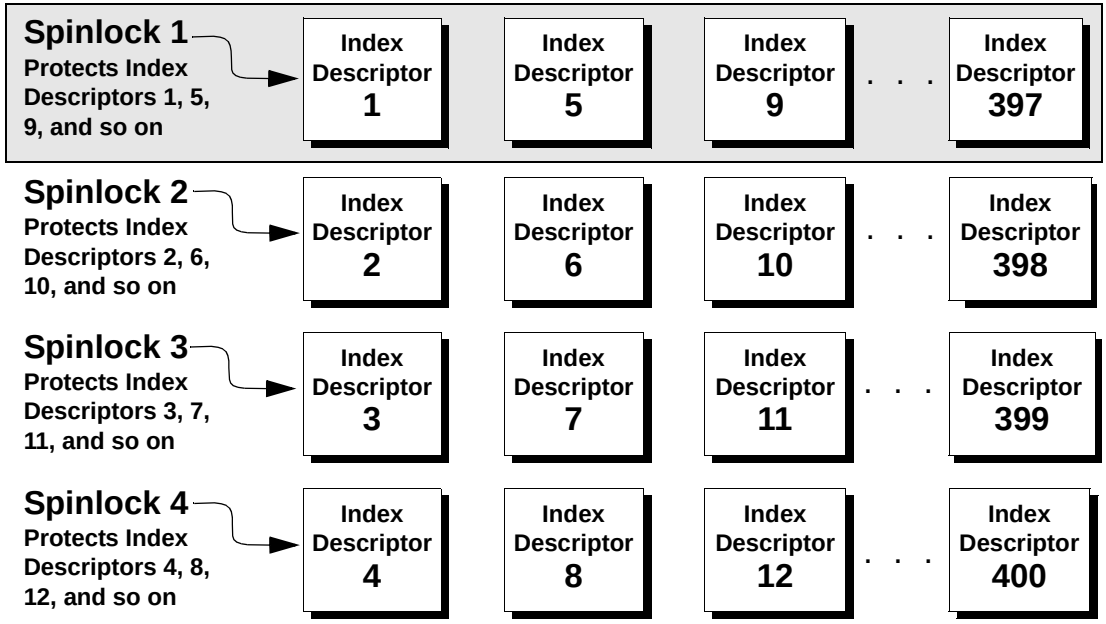
- Round-robin assignment
- Sequential assignment

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 5-2 illustrates one example of the round-robin assignment method and shows the relationship between spinlocks and index metadata descriptors.

Figure 5-2: 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 assigns 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 the *Performance and Tuning Guide* for information on spinlock contention.

CHAPTER 6

Creating and Managing User Databases

This chapter explains how to create and manage user databases.

Commands for creating and managing user databases

Table 6-1 summarizes the commands for creating, modifying, and dropping user databases and their transaction logs.

Table 6-1: 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. 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.
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 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 users.

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 sp_changedbowner, as described in "Changing database ownership" on page 148.

When a System Administrator grants permission to create databases, the user that receives the 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 for alter database or drop database defaults to the Database Owner, and permission is automatically transferred with database ownership. You cannot use grant or revoke to change alter database and drop database permission.

Using the *create database* command

Use create database to create user databases. You must have create database permission, and you must be a valid user of master. Always type use master before you create 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 13, "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 | default_location = "pathname"}]
    [for {load | proxy_update}]
```

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 141 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 143 for more information.

- for load causes Adaptive Server to skip the page-clearing step during database creation. 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 147 for more information.
- with override allows Adaptive Servers on machines with limited space to maintain their logs on device fragments that are separate 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 148 for more information.
- *size* is in the following unit specifiers: 'k' or 'K' (kilobytes), 'm' or 'M' (megabytes), and 'g' or 'G' (gigabytes), 't' or 'T' (terabytes). If you do not specify the unit size, create database assumes 'M', megabytes.

How *create database* works

When a user with the required permission issues create database, Adaptive Server:

- Verifies that the database name specified is unique.
- Makes sure that the database device names specified 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 to load a database dump, for load skips page clearing, which 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, including:

- 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 databases to inherit particular options, change the options in model with sp_dboption. See Chapter 2, "System and Optional Databases," for more information about model. See Chapter 8, "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 Chapter 13, "Getting Started With Security Administration in Adaptive Server," 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. create database 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 You can 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 143 for more information.

If you use the default keyword, or if you omit the on clause, Adaptive Server puts the database on one or more of the default database devices specified in master..sysdevices. See "Initializing Database Devices" on page 245 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 = "4M"
```

To place the database on specific database devices, give the names of the database devices where you want it stored. 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 7, "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. The database and transaction log are not separated:

```
create database newdb
on mydata = "3M", newdata = "2M"
```

Warning! Unless you are creating a small or noncritical database, always place the log on a separate database device. Follow the instructions in "Placing the transaction log on a separate device" on page 143 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 and 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, create database fails.

If you create (or alter) a database on a UNIX device file that does not use the dsync setting, Adaptive Server displays an error message in the error log file. For example, if you create the "mydata" device in the previous example does not use dsync, you see a message similar to:

```
Warning: The database 'newdb' is using an unsafe virtual device 'mydata'. The
recovery of this database can not be guaranteed.
```

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 smallest database you can create is the size of the model database, which is determined by your installation's logical page size. If you want to increase the minimum size of a database, use alter database to enlarge the model database. You can also use the default database size configuration parameter to determine the default database size. See Chapter 5, "Setting Configuration Parameters," for more information.

If you omit the on clause, the database is created as 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.

To see the logical names of default database devices, enter:

```
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 size allocation decisions you make are important, because it is difficult to reclaim storage space after 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 sp_estspace or by calculating the value. See Chapter 15, "Determining Sizes of Tables and Indexes," in the *Performance and Tuning Guide: Monitoring* for instructions.

Placing the transaction log on a separate device

Use the log on clause of the create database command to place the transaction log (the syslogs table) on a separate database device. Unless you are creating very small, noncritical databases, always place the log on a separate database device. Placing the logs on a separate database device:

- Lets you use dump transaction, 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 15, "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.

To specify a size and device for the transaction log, use the log on *device* = *size* clause to create database. The size is in the unit specifiers 'k' or 'K' (kilobytes), 'm' or 'M' (megabytes), and 'g' or 'G' (gigabytes), 't' or 'T' (terabytes). 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 = "8M", newdata = "4M"
log on tranlog = "3M"
```

Estimating the transaction log size

The size of the transaction log is determined by:

- 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 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 also helps you choose thresholds for the log and scheduling the timing of transaction log dumps. To check the space used by a database's transaction log, first use the database. Then enter:

```
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 quickly the log grows.

Default log size and device

If you omit the *size* parameter from the log on clause, Adaptive Server locates the minimum permitted amount of storage. If you omit the log on clause entirely, Adaptive Server places the 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 use sp_logdevice to move your transaction log to another database device.

sp_logdevice marks the portions of an existing database that exist on a specified device as reserved for the transaction log; it does not move existing data. If your database already has data on this device, Adaptive Server does not interpret this data as not being on its proper segment. However, because dbcc reports this as an error, no existing part of the log moves to the specified device; the current log data remains where it is until the log has extended onto the new device and you use dump transaction to clear that part of the log. Also, sp_logdevice does not allocate new space to the database or initialize devices. Instead, it reserves those portions of the specified device for the log that already belong to the database you specify.

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:

1. Execute sp_logdevice, naming the new database device.
2. Execute enough transactions to fill the page that is currently in use. The amount of space you need to update depends on the size of your logical pages. 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. You may want to put the database into single-user mode with sp_dboption.
4. Run dump transaction, which 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 are removed. See Chapter 11, "Developing a Backup and Recovery Plan," for more information.

5 Run 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 11, "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, and creates a target database that can be used *only* for loading a dump.

If you create a database using for load, 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 must match the usage allocations in the dumped database. See Chapter 12, "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 machines with limited space to maintain their logs on device fragments that are separate from their data. Use this option if you must 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 must get databases back online following a hard disk crash.

You can still dump your transaction log, but if you experience a media failure, you cannot access the current log, since it is on the same device as the data. You can recover only to the last transaction log dump, and all transactions between that point and the failure time are lost.

In the following example, the log and data are on separate fragments of the same logical device:

```
create database littledb
    on diskdev1 = "4M"
    log on diskdev1 = "1M"
    with override
```

The minimum database size you can create is the size of model.

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. sp_changedbowner changes the ownership of a database. The procedure must be executed by the System Administrator in the database where the ownership is to 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 Chapter 13, "Getting Started With Security Administration in Adaptive Server," 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 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 alter database 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 for alter database defaults to the Database Owner, and is automatically transferred with database ownership. For more information, see "Changing database ownership" on page 148. alter database permission cannot be changed with grant or revoke.

Note alter database for proxy update drops all proxy tables in the proxy database.

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]
[for proxy_update]
```

In its simplest form, alter database adds the configured default amount of space 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 space from a default database device to the newpubs database, enter:

```
alter database newpubs
```

The on and log on clauses operate like the corresponding clauses in create database. 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 or one allocation unit, whichever is larger.

To add 3MB to the space allocated for the newpubs database on the database device named pubsdata1, enter:

```
alter database newpubs
on pubsdata1 = "3M"
```

If Adaptive Server cannot allocate the requested size, it allocates as much as it can on each database device, with a minimum allocation of 256 logical 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 = "2M", pubsdata2 =" 3M"
log on tranlog
```

Note Each time you issue the alter database command, dump the master database.

Use with override 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.

Use for load 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 12, "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 drop database 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.

The syntax 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

To create a database on a database device and allocate a certain amount of space to it, Adaptive Server first makes an entry for the new database in sysdatabases. Then, it checks master..sysdevices to make sure that the device names specified in create database 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. The database storage space can even be drawn from more than one database device. 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. 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 rest of the pages are used.

The *sysusages* table

The database storage information is listed in 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 you install Adaptive Server, sysusages contains rows for these databases:

- master, with a dbid of 1
- The temporary database, tempdb, with a dbid of 2
- model, with a dbid of 3
- sybsystemdb, with a dbid of 31513
- sybsystemprocs, with a dbid of 31514

If you upgraded Adaptive Server from an earlier version, databases sybsystemdb and sybsystemprocs may have different dbids.

If you installed auditing, the sybsecurity database is dbid 5.

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 that includes the five system databases and one user database. The user database was created with the `log on` option, and was extended once using alter database. It has dbid 4:

```
select dbid, segmap, lstart, size, vdevno, vstart
from sysusages
order by 1
dbid      segmap      lstart      size        vdevno      vstart
----      ---------   ----------  --------    --------    -----------
1         7           0           6656        0           4
2         7           0           2048        0           8196
3         7           0           1536        0           6660
4         3           0           5120        2           0
4         4           5120        2560        3           0
4         3           7680        5120        2           5120
31513     7           0           1536        0           10244
31514     7           0            63488       1            0
```

In this example, the lstart and size columns describe logical pages whose size may vary from 2k to 16k bytes. The vstart column describes virtual pages (whose size is always 2k bytes). These global variables show page size information:

- *@@maxpagesize* – logical page size
- *@@pagesize* – virtual page size

The following matches the database ID to its name, shows the number of megabytes represented by the size column, shows the logical device name for each vdevno in the list, and computes the total number of megabytes allocated to each database. The example output shows only the result for dbid 4, and the result has been reformatted:

```
select  dbid, db_name(dbid) as 'database name', lstart,
     size / (power(2,20)/@@maxpagesize) as 'MB',
     d.name
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and d.status & 2 = 2
order by 1
compute sum(size / (power(2,20)/@@maxpagesize)) by dbid
dbid     database name     lstart      MB          device name
------    --------------    --------    --------    --------------------
4         test              0           10          datadev
```

```
4          test                5120         5            logdev
4          test                7680         10           datadev

Compute Result:
-----------
25
```

The following describes the changes to the segmap values in the sysusages table as you add segments. The server in the example initially includes the default databases and a user database named testdb (a data-only database), and a log on the testlog device, as shown in the following output from the sysusages table:

```
select dbid, segmap from master..sysusages where dbid = 6
 dbid    segmap
------  -----------
  6           3
  6            4
```

If you add a user segment newseg to the test database and create table abcd on newseg and again select the segment information from sysusages:

```
sp_addsegment newseg, testdb, datadev

create table abcd ( int c1 ) on newseg

select dbid, segmap from sysusages
where dbid=6
dbid     segmap
------   -----------
6           11
6            4
```

The segment mapping for the user database has changed from a value of 3 to a value of 11, which shows that segments mappings for user databases are not permanent, but change when you reconfigure a database

Run sp_helpsegment to determine the status of the segments:

```
sp_helpsegment
 segment     name             status
-------    ---------------- ----------
0           system              0
1           default             1
2           logsegment          0
3           newseg              0
```

The segment newseg is not part of the default pool.

If you add another segment, newseg1, to the testdb database and select the segment information from sysusages again, the segment mapping for newseg has changed from 11 to 27:

```
sp_addsegment newseg1, testdb, datadev

select dbid, segmap from sysusages
 dbid    segmap
------  -----------
6             27
6               4
```

The master..sysusages segmap column is a bit map of segment assignments. segmap shows the storage that is permitted for the database fragment it represents. You control the bit values in this mask using stored procedures for segment management. The valid bit's numbers in the mask come from syssegments in the local database. (Your "local" database is the database you are currently using: either your default database from login, or the database you most recently used with use database.)

Adaptive Server supplies three named segments:

- system, which is segment 0
- default, which is segment 1
- logsegment, which is segment 2

Use sp_addsegment to create additional segments. If you create segments in the model database, these segments exist in all databases you subsequently create. If you create them in any other database, they will exist only for that database. Different segment names in different databases can have the same segment number. For example, newseg1 in database testdb and mysegment in database mydb can both have segment number 4.

See below for more information about the possible values for the segmap column.

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:

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 8, "Creating and Using Segments."

The query below illustrates the connection between segments in syssegments and segmap in master..sysusages. The query lists the segment mappings for the current database, showing that each segment number in syssegments becomes a bit number in master..sysusages:

```
select dbid, lstart, segmap, name as 'segment name'
from syssegments s, master..sysusages u
where u.segmap & power(2,s.segment) != 0
and dbid = db_id()
order by 1,2
dbid   lstart    segmap       segment name
-----   ------    --------     --------------
4       0         3           system
4       0         3           default
4       5120      4           logsegment
4       7680      3           system
4       7680       3           default
```

This example shows that disk fragment for lstart value 0 and the fragment for lstart value 7680 use segments system number 0 and default number 1, while the fragment for lstart value 5120 uses segment logsegment number 2. This database was created using both the on and log on clauses of create database, and was then extended once using the on clause of alter database.

Because the sysusages segmap uses an int datatype, it can contain only 32 bits, so no database can hold more than 32 segments (numbered 0 - 31). Because segmap is a signed quantity (that is, it can display both positive and negative numbers), segment 31 is perceived as a very large negative number, so the query above generates an arithmetic overflow when you use it in a database that uses segment 31.

The *lstart*, *size*, and *vstart* columns

- lstart column – 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 – the number of contiguous 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 – the address where the piece assigned to this database begins.
- vdevno – the device in which this database fragment resides.

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 sp_helpdb with the database name:

```
                    sp_helpdb pubs2
name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa           4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data

device_fragments    size          usage         created      free kbytes
------------------- ------------ ------------- ----------   ------------
master              10.0MB        data and log  Apr 13 2005          1792

pubs_2_dev         10.0MB        data and log  Apr 13 2005         9888

device                  segment
----------------------- -----------------------
master                  default
master                  logsegment
master                  system
```

```
pubs_2_dev              default
pubs_2_dev              logsegment
pubs_2_dev              system
pubs_2_dev              seg1
pubs_2_dev              seg2
```

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 8, "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 8, "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          48.0 MB   sa      1   Apr 12, 2005       mixed log and data
model            8.0 MB   sa      3   Apr 12, 2005       mixed log and data
pubs2           20.0 MB   sa      6   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
sybsystemdb      8.0 MB   sa      5   Apr 12, 2005       mixed log and data
sybsystemprocs 112.0 MB   sa      4   Apr 12, 2005       trunc log on chkpt,
     mixed log and data
tempdb           8.0 MB   sa      2   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
```

Checking the amount of space used

sp_spaceused provides a summary of space use:

- In the database
- By a table and its indexes and text/image storage
- 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 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 6-2 describes the columns in the report.

Table 6-2: 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	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	The amount of space used by data and indexes.
unused	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 equal 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:

```
sp_spaceused titles
name   rowtotal reserved  data    index_size unused
------ -------- --------- ------- ---------- -----
titles 18       48 KB     6 KB    4 KB       38 KB
```

The rowtotal column may be different than 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. update statistics, dbcc checktable, and dbcc checkdb update the rows-per-page estimate, so rowtotal is most accurate after you have run one of these commands has been run.

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(convert(numeric(20,0), high - low + 1))
from sysdevices
where status & 2 = 2

-----------------
             230224
```

In this example, the 2 in the status column indicates a physical device. high is the highest valid 2k block on the device, so you must add 1 to get the true count from the subtraction and convert counts to numeric(20,0) to avoid overflow from integer addition in the sum.

CHAPTER 7

Database Mount and Unmount

The mount and unmount commands allow you to transport databases from a source Adaptive Server to a destination Adaptive Server.

Overview

The key benefits of the mount and unmount commands are:

1 The copy operation provided by the quiesce and mount commands enables you to package proprietary databases more easily; for example, data files instead of SQL scripts. The associated actions necessary for running these SQL scripts, such as device and database setup, are eliminated.

2 The quiesce database extension for mount allows you to copy databases without a shutdown of the Adaptive Server. The primary Adaptive Server is the source, and the secondary Adaptive Server is the destination. quiesce database also allows a single secondary Adaption Server to act as standby for databases from multiple primaries, since databases from multiple sources can be copied to a single destination.

The mount and unmount commands support two operations:

- Copying a database – When you copy a database you must use a command outside of Adaptive Server, such as UNIX dd or ftp, to create a byte-for-byte copy of all pages in a set of one or more databases

- Moving a database – When you move a set of databases from a source Adaptive Server to a destination Adaptive Server, you are physically moving the underlying devices.

With the mount and unmount commands:

1 Remove a database using unmount, which removes a database and its devices from a server. A manifest file is created for the database that you are unmounting at a location you specify in the command clauses. The manifest file contains information pertinent to the database at the source Adaptive Server, such as database devices, server information, and database information. For more information see "Manifest file" on page 164.

2 Copy or move the database onto the destination Adaptive Server, then use mount to add the devices, attributes, and so on for the database.

3 Use database online to bring the database up on the destination Adaptive Server without restarting the server.

Warning! For every login that is allowed access to a database on the original Adaptive Server, a corresponding login for the same suid must exist at the destination Adaptive Server.

For permissions to remain unchanged, the login maps at the destination Adaptive Server must be identical to those on the source Adaptive Server.

Manifest file

The manifest file is a binary file that contains information pertinent to the database, such as database devices, server information, and database information. Both the source and destination Adaptive Servers must use the same version of the manifest file. The manifest file can be created only if the set of databases that occupy those devices are isolated and self-contained. The manifest file contains:

- Source server information that is server-specific or common to all the databases, such as the version of the source Adaptive Server, any upgrade version, page size, character set, sort order, and the date the manifest file was created.

- Device information that is derived from the sysdevices catalog. The manifest file contains the information on the first and last virtual pages, the status, and the logical and the physical names of the devices involved.
- Database information that is derived from the sysdatabases catalog. The manifest file contains information on the database name, dbid, login name of the database administrator (dba), suid of the owner, pointer to the transaction log, creation date, status bits from the status fields in sysdatabases, date of the last dump tran, and the diskmap entries of the databases involved.

Warning! The manifest file is a binary file, so operations that perform character translations of the file contents (such as ftp) corrupt the file, unless performed in binary mode.

Considerations

There are several things you should consider in using the mount and unmount commands.

Device allocation

Moving or copying are database-level operations that require activity external to Adaptive Server. To move or copy devices and databases, be sure that they are set up on the source Adaptive Server in units supporting a physical transportation.

For example, if any device is used by more than one database, then all of those databases must be transported in one operation.

Moving a database

Moving a set of one or more databases means physically moving the underlying devices. unmount the databases at the source, create the manifest file, and move the devices to the destination Adaptive Server.

At the destination Adaptive Server, mount the database using the manifest file. To activate the database, use the online database command. You need not reboot the destination server.

Use caution during the initial configuration of the source Adaptive Server. The Adaptive Server cannot know if a device is transportable as a unit. Make sure that the underlying disk that is to be disconnected does not cause a database that is not being moved to lose some of its storage. Adaptive Server cannot identify whether a drive is partitioned on a single physical disk; you must move the databases together in one unit.

Warning! mount and unmount allow you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

Copying a database

Copying is duplicating a set of databases from the source to a destination by physically copying the underlying devices. In this case, you are copying a set of databases from a source Adaptive Server to a destination Adaptive Server.

The quiesce database for an external dump includes the extension for creating the manifest file. Then use a utility or command external to Adaptive Server, (tar, zip or the UNIX dd command) to move or copy the database to the destination Adaptive Server. The data dump and the manifest file are then moved to the destination Adaptive Server. Data is extracted and placed on devices at the destination Adaptive Server by the tar, unzip, or dd command.

If a device is used for more than one database, all of the databases on that device must be moved in one operation.

Performance considerations

- Database IDs for the transported databases must be the same on the destination Adaptive Server; otherwise, mount displays a message asking you to run dbcc checkalloc on the database.

  You must run checkalloc to fix up the database ID if the database is not being mounted for temporary usage.

- If the dbid is changed, all stored procedures are marked for recompiling in the database. This increases the time taken to recover the database at the destination, and delays the first execution of the procedure.

System restrictions

- You cannot unmount system databases. However, you can unmount sybsystemprocs.
- You cannot unmount proxy databases or user-created temporary databases.
- You cannot use the mount and unmount commands in a transaction.
- You cannot mount a database on a server configured for high availability.
- A mount or unmount command limits the number of databases to eight in a single command.

Configuration restrictions

When executing the mount command at the destination Adaptive Server, use these rules:

- The page size in the destination Adaptive Server must be equal to the page size in the source Adaptive Server.
- There must be enough devices configured at the destination Adaptive Server to successfully add all the devices belonging to the mounted databases.
- You cannot have databases and devices with the same names on the destination Adaptive Server as those being mounted from the source Adaptive Server.
- The destination Adaptive Server and the source Adaptive Server must use the same version as the manifest file.
- The platforms of the source and destination Adaptive Servers must be the same.
- Differences in the sort order or character set are resolved by rules followed by load database. A database with a different character set can be mounted only if the sort order is binary.

Logical restrictions

You cannot mount a subset of a transported set of databases at the destination Adaptive Server; all databases and their devices in the manifest file must be mounted together.

A set of databases copied to a destination Adaptive Server cannot contain any references to databases outside of the set.

Device verification

The destination Adaptive Server verifies the set of devices provided in the manifest file by scanning the device allocation boundaries for each database. The scan ensures that the devices being mounted correspond to the allocations described in the manifest file and verifies the dbid in the allocation page against that in the manifest file for the first and last allocation pages for each *sysusages* entry.

If a stricter device inspection is necessary, you can use the with verify in the mount command, which verifies the dbid for all allocation pages in the databases.

You must exercise extreme care so as not to mix up copies of the devices.

For example, if you make a database copy made up of copies for disks dsk1, dsk2 and dsk3 and in August, you try to mount dsk1 and dsk2 copies from a copy of the database made in March, and dsk3 from a copy made in June, the allocation page check passes even if with verify is used in the mount command. The database recovery fails since the information is not valid for the version being used.

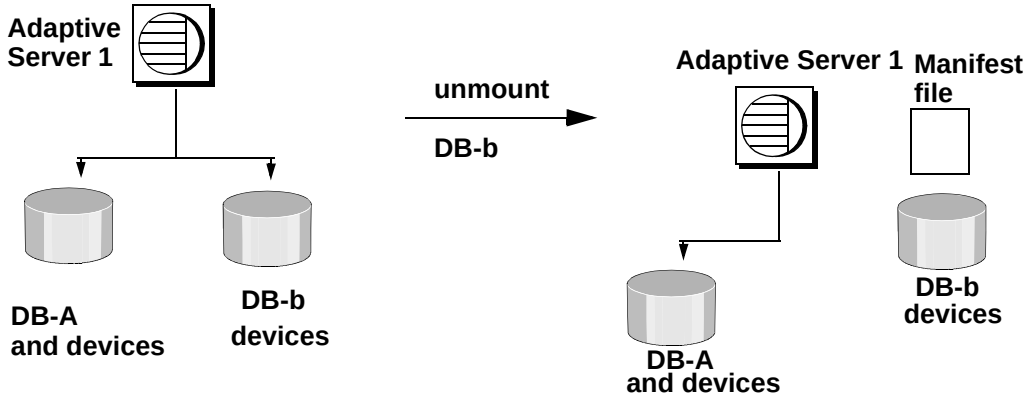
However, recovery may not fail if dsk3 is not accessed, which means that the database comes online, but the data could be corrupt.

Commands

This section explains how the mount and unmount commands are used. The quiesce database command includes a clause that facilitates the mount and unmount commands.

Unmounting a database

Figure 7-1: unmount command



Note unmount allows you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

When you unmount a database, you remove the database and its devices from an Adaptive Server. The unmount command shuts down the database. All tasks using the database are terminated. The database and its pages are not altered and remain on the operating system devices.

The unmount command limits the number of databases that can be moved in a single command to eight.

The unmount command:

- Shuts down the database,
- Drops the database from the Adaptive Server,
- Deactivates and drops devices,
- Uses the *manifest_ file* extension to create the manifest file.

Once the unmount command completes, you can disconnect and move the devices at the source Adaptive Server if necessary.

```
unmount database <dbname list> to <manifest_file> [with
{override, [waitfor=<delay time]} ]
```

For example:

```
1> unmount database pubs2 to
"/work2/Devices/Mpubs2_file"
2> go
```

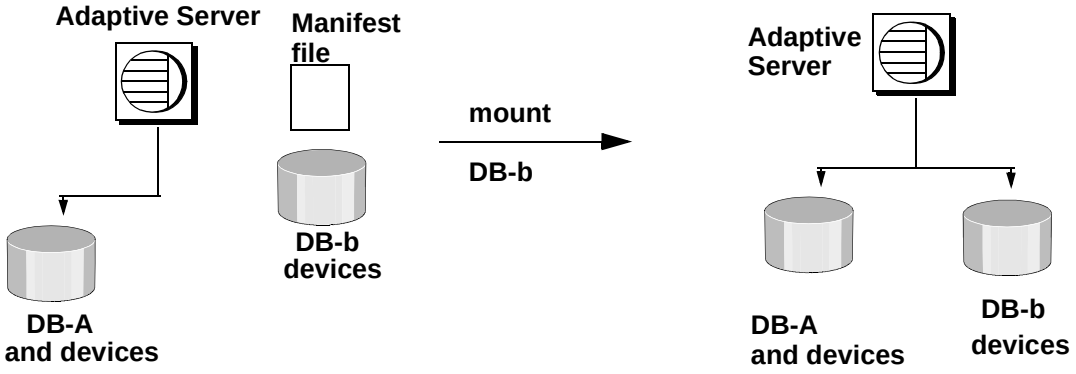
If you now try to use the pubs2 database, you see this error:

```
Attempt to locate entry in sysdatabases for database
'pubs2' by name failed - no entry found under that name.
Make sure that name is entered properly.
```

Note When the referencing database is dropped by the unmount command with an override, you cannot drop the referential constraints (dependencies) or table.

Mounting a database

Figure 7-2: mount command



Use the mount command to attach the database to the destination or secondary Adaptive Server. The mount command decodes the information in the manifest file and makes the set of databases available online. All the required supporting activities are executed, including adding database devices, if necessary, and activating them, creating the catalog entries for the new databases, recovering them, and putting them online.

The mount command limits the number of databases to eight in a single command.

Note mount allows you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

You can use the mount command in different ways:

- When you are ready, use the mount command at the destination Adaptive Server:

  mount database all from <*manifest_file*> [with {verify, listonly}] [using *device_specifications*]

  device_specifications ::=
  ”physicalname”[=”device_name”][,”physicalname”[‘device_name]]

  For example:

  ```
  1> mount database all from "/data/sybase2/mfile1"
  using "/data/sybase1/d0.dbs" = "1dev1"

  2> go
  ```

  The databases and their devices appear at the destination Adaptive Server, marked as *in-mount*. The system is populated with updates to system catalogs and appropriate information on the databases, but the databases themselves are unrecovered. They do, however, survive a system crash.

  The destination Adaptive Server then recovers the databases one at a time. The databases are left offline after recovery.

  If a recovery fails on a database, it affects only that database. The recovery continues for the other databases.

  Use the command database online to bring the databases online.

  You need not restart the destination server.

- Using the mount command with listonly displays the path names in the manifest file from the source Adaptive Server without mounting the database.

  Before mounting the databases, use the listonly parameter to list the device path names at the destination Adaptive Server. Then use the mount command to actually mount the databases.

mount database all from <*manifest_file*> with listonly

As an example:

```
1> mount database all from "/data/sybase2/mfile1"
with listonly
2> go

/data/sybase1/d0.dbs = ldev1
```

Once you have the path names, you can verify or modify them to meet your criteria at the destination Adaptive Server.

Renaming devices

The manifest file contains the device paths as known to the source Adaptive Server that created the manifest file. If the destination Adaptive Server accesses the devices with a different path, you can specify the new path to the mount command.

1 Use mount with listonly to display the old path:

```
1> mount database all from "/data/sybase2/mfile1"
with listonly
2> go

"/data/sybase1/d0.dbs" = "1dev1"
```

2 If the new path for the device `"/data/sybase1/d0.dbs"` is `"/sybase/devices/d0.dbs"`, specify the new device in the mount command:

```
mount database all from "/data/sybase2/mfile1" using
"/sybase/devices/d0.dbs" = "ldev1""
```

Destination changes

Once databases are mounted on the destination Adaptive Server, certain settings are cleared on the mounted database:

- Replication is turned off.
- Audit settings are cleared and turned off.
- CIS options, default remote location, and type are cleared.
- Cache bindings are dropped for both the mounted databases and their objects.

- Recovery order is dropped for the mounted databases and becomes the default dbid order.

quiesce database extension

To duplicate or copy databases, use quiesce database with the extension for creating the manifest file. quiesce database affects the quiesce hold by blocking writes in the database, and then creates the manifest file. The command then returns control of the database to the user.

You can now use a utility to copy the database to another Adaptive Server. These rules for quiesce database hold must be followed for the copy operation:

- The copy operation cannot begin until the quiesce database hold process has completed.
- Every device for every database in the quiesce database command must be copied.
- The copy process must complete before you invoke quiesce database release.

Syntax

quiesce database *tag_name* hold *dbname_list* [for external dump]
[to *manifest_file* [with override]]

Parameters

Where:

- *tag_name* – is a user-defined name that designates the list of databases to hold or release. *tag_name* must conform to the rules for identifiers.
- *dbname_list* – the list of the databases included in the quiesce database hold command.
- for external dump – specifies that while updates to the databases in the list are suspended, you physically copy all affected database devices, using some facility external to Adaptive Server. The copy operation serves as a replacement for the combination of dump database and load database.
- *manifest_file* – is the binary file that describes the databases included in the command. It can be created only if those databases are isolated and self-contained on their devices.

Examples

Example 1

Place the database in a hold status and build the manifest file using the quiesce command:

```
quiesce database pubs2_tag hold pubs2 for external dump
to "/work2/Devices/Mpubs_file" with override
```

Once the command completes, control returns to the user.

Example 2

Copy the database devices.

If you do not know which devices are to be copied, use the mount database with listonly to list all the devices to be copied.

```
1> mount database all from "/data/sybase2/mfile1" with
listonly
2> go

"/data/sybase1/d0.dbs" = "1dev1"
```

You cannot create a manifest file if the set of databases that are quiesced contain references to databases outside of the set. You may use with override option to bypass this restriction.

```
quiesce database pubs2_tag release for external dump to
Mpubs_file
```

CHAPTER 8

Creating and Using Segments

This chapter introduces the system procedures and commands for using segments in databases.

See also Chapter 6, “Controlling Physical Data Placement,” in the *Performance and Tuning Guide: Basics* for information about how segments can improve system performance.

What is a segment?

A segment is 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 improve 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 176). 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.

System-defined segments

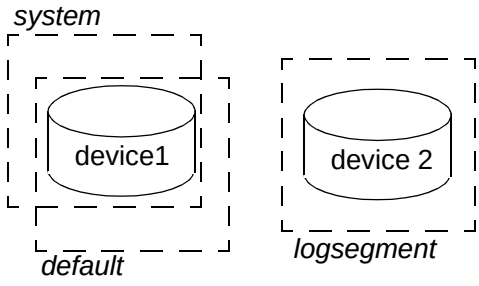
When you first create a database, Adaptive Server creates three segments in the database, as described in Table 8-1.

Table 8-1: 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 8-1.

Figure 8-1: 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 8-2 summarizes the commands and system procedures for managing segments.

Table 8-2: 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 partition 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 6, "Creating and Managing User Databases," for examples.
sp_help	Displays information about a table, including the segment where the table resides.
sp_helpindex	Displays information about a table's indexes, including the segments where the indexes reside.

Why use segments?

When you add a new device to a 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.

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.

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 180 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.

If you create additional segments for data, you can create new threshold procedures for each segment. See Chapter 15, "Managing Free Space with Thresholds," for more information.

Improving performance

In a large, multidatabase or multidrive Adaptive Server environment, you can enhance system performance by paying careful attention to the allocation of space to databases and the placement of database objects on physical devices. 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. The *Performance and Tuning Guide: Basics* also offers more information about how segments can improve performance.

Separating tables, indexes, and logs

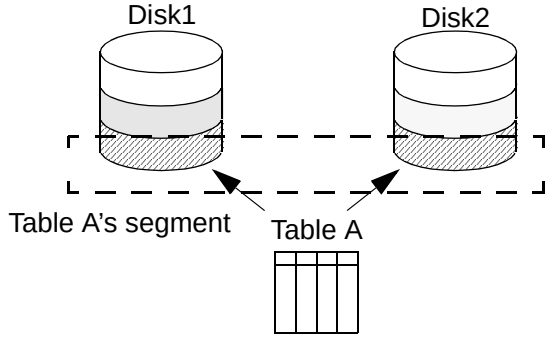
Generally, placing a table on one physical device, its nonclustered indexes on a second physical device, and the transaction log on a third physical device can speed performance. Using separate physical devices (disk controllers) reduces the time required to read or write to the disk. If you cannot devote 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) places 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 183 for information about placing tables and indexes on segments.

Splitting tables

You can split a large, heavily used table across devices on separate disk controllers to 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 8-2 shows a table that is split across the two devices in its segment.

Figure 8-2: 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 has a clustered index, use partial loading.
- If the table contains text or image datatypes, separate the text chain from other data.

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 8-2). Partitioning a table increases both insert 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 Chapter 6, "Controlling Physical Data Placement," in the *Performance and Tuning Guide: Basics* for information about partitioning tables using alter table.

Partial loading

To split a table with a clustered index, 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 186 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 189 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 190 for more information and syntax.

Creating segments

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.

Once the database device exists and is available to the database, define the segment in the database with sp_addsegment. The syntax is:

```
sp_addsegment segname, dbname, devname
```

where:

- segname is any valid identifier. 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

When you use segments, you also need to manage their scope—the number of database devices to which each 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. 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,
- The database device must be available in the desired database, and
- The segment name must exist in the current database.

The following example adds the database device pubs_dev2 to an existing segment named bigseg:

```
sp_extendsegment bigseg, pubs2, pubs_dev2
```

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 = "2M"
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 = "2M"
```

See "A segment tutorial" on page 195 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, reduce the scope of the default and system segments so that they no longer point to the new device.

Use sp_dropsegment to drop a single database device from a segment, reducing the segment's scope:

sp_dropsegment *segname*, *dbname*, *device*

With three arguments, sp_dropsegment 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 190.

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 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

Creating new objects on segments

To place a new object on a segment, first 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. 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 create table or create index with the optional on *segment_name* clause to create the object on the segment. The syntax is:

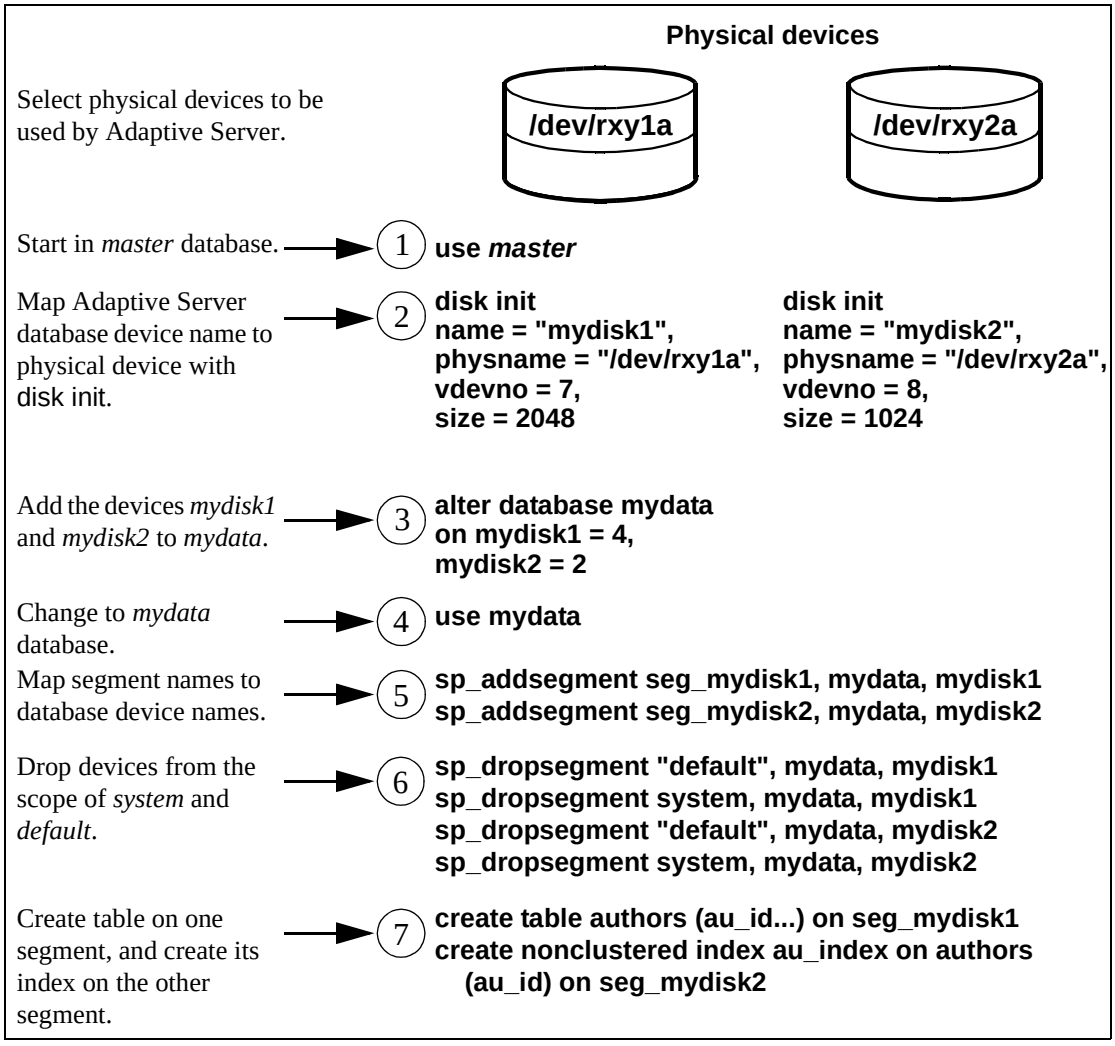
```
create table table_name (col_name datatype ... )
    [on segment_name]

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 190.

Example: creating a table and index on separate segments

Figure 8-3 summarizes the sequence of Transact-SQL commands used to create tables and indexes on specific physical disks on a server using 2K logical page size.

Figure 8-3: Creating objects on specific devices using segments



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 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.

Placing existing objects on segments

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
```

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.

Note 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 190 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.

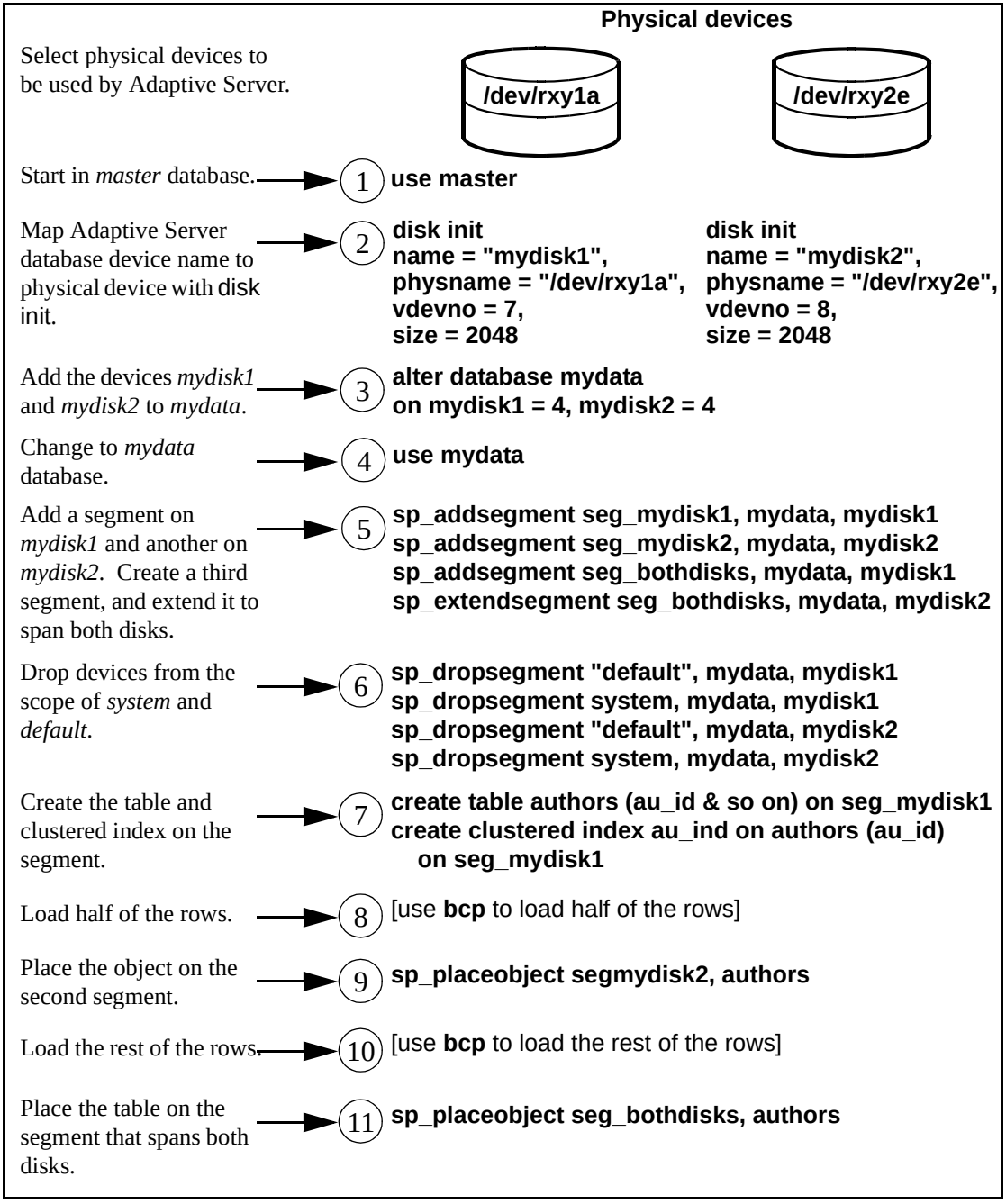
Example: splitting a table and its clustered index across physical devices

Performance can be improved for high-volume, multiuser applications when large tables are split across segments that are located on separate disk controllers.

The order of steps is quite important at certain stages. In particular, you must create the clustered index before you place the table on the second segment.

Figure 8-4 summarizes the process of splitting a table across two segments on a server using a 2K logical page size:

Figure 8-4: Splitting a large table across two segments



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. Use sp_placeobject again to place the table on the segment that spans both devices.

The balance of disk allocation may change over time if the table is updated frequently. To guarantee that the speed advantages are maintained, you may need to drop and re-create the table at some point.

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 with 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.

If you want the text pages to be on a particular segment, first create the table on that segment (allocating the initial extent on that segment), then create a clustered index on the table to move the rest of the data to the 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 its clustered index on a different segment, the table migrates to the segment where you created the clustered index. 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 199 for an example of this command.

Dropping segments

When you use 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 segments or drop the objects first and then drop the segment.

The syntax for dropping a segment is:

```
sp_dropsegment segname, dbname
```

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, however, reduce the scope of these segments—see "Reducing the scope of a segment" on page 183.

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

These system procedures provide information about segments:

- sp_helpsegment – lists the segments in a database 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

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, you see information about the segments in the database.

For example:

```
sp_helpdb pubs2

name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa           4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data
```

```
device_fragments    size          usage         created      free kbytes
------------------- ------------- ------------- ----------   ------------
master              10.0MB        data and log  Apr 13 2005          1792
pubs_2_dev         10.0MB        data and log  Apr 13 2005          9888

device                  segment
----------------------- ----------------------
master                  default
master                  logsegment
master                  system
pubs_2_dev              default
pubs_2_dev              logsegment
pubs_2_dev              system
pubs_2_dev              seg1
pubs_2_dev              seg2
```

sp_help and *sp_helpindex*

When you execute sp_help and sp_helpindex in a database, and give a table's name, you see information about which segments store the table or its indexes.

For example:

```
sp_helpindex authors
```

```
index_name  index_keys  index_description  index_max_rows_per_page
     index_fillfactor  index_reservepagegap  index_created
     index_local
----------  ----------  -----------------  ------------------------
     ----------------  --------------------  -------------
     -----------
auidind     au_id       clustered,unique                          0
                    0                    0   Apr26 2005  4:04PM
     Global Index
aunwind    au_lname,au_fname  nonclustered,unique               0
                    0                    0   Apr26 2005  4:04PM
     Global Index
(2 rows affected)
index_ptn_name        index_ptn_seg
--------------------  --------------
auidind_400001425     default
aunmind_400001425     default
```

Segments and system tables

Three system tables store information about segments: master..sysusages and two system tables in the user database, sysindexes and syssegments. sp_helpsegment uses these tables. Additionally, 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 partition, Adaptive Server adds a new row to sysindexes. The segment column in that table stores the segment number, showing where the server allocates 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 195.

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 152 for more information about the segmap column and the system tables that manage storage.

A segment tutorial

The following tutorial shows how to create a user segment and how to remove all other segment mappings from the device. The examples in this section assume a server using 2K logical page sizes.

When you are working with segments and devices, remember that:

- If you assign space in fragments, each fragment has an entry in sysusages.
- When you assign an additional fragment of a device to a database, 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 = "5M"
     log on logdev = "4M"
```

Now, if you use mydata, and run sp_helpdb, you see:

```
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       9.0 MB sa             5 May 27, 2005  no options set

device_fragments  size    usage      created              free kbytes
----------------  ------  ---------- -----------          -----------
bigdevice         5.0 MB  data only  May 25 2005 3:42PM   3650
```

```
logdev            4.0 MB  log only   May 25 2005 3:42PM   not applicable
------------------------------------------------------------------------
log only free kbytes = 4078
device                  segment
----------------------- ----------------------
bigdevice               default
bigdevice               system
logdev                  logsegment
(return status = 0)
```

Like all newly created databases, mydata has the segments named default, system, and logsegment. Because create database 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 run sp_helpdb again, you see entries for the added fragments:

```
use master
alter database mydata on bigdevice = "2M"
    log on logdev = "1M"
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created       status
---------- -------- --------- ------ ------------  ---------------
mydata     12.0 MB  sa             4 May 25, 2005  no options set

device_fragments  size     usage      created                 free kbytes
----------------- -------- ---------- ------------------      -----------
bigdevice         5.0 MB   data only  May 25 2005 3:42PM            2048
logdev            4.0 MB   data only  May 25 2005 3:42PM     not applicable
data only         2.0 MB   log only   May 25 2005 3:55PM            2040
log only          1.0 MB   log only   May 25 2005 3:55PM     not applicable
------------------------------------------------------------------------
log only free kybytes = 5098
device                  segment
----------------------- ----------------------
bigdevice               default
bigdevice               system
logdev                  logsegment
```

Always add log space to log space and data space to data space. Adaptive Server instructs 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.

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      15.0 MB sa             5 May 25, 2005  no options set

device_fragments   size      usage       created             free kbytes
------------------ --------- ----------- ------------------  --------------
bigdevice          5.0 MB    data only   May 25 2005 3:42PM            3650
logdev             4.0 MB    log only    May 25 2005 3:42PM  not applicable
bigdevice          2.0 MB    data only   May 25 2005 3:55PM            2040
logdev             1.0 MB    log only    May 25 2005 3:55PM  not applicable
newdevice          3.0 MB    data only   May 26 2005 11:59AM           3060
------------------------------------------------------------------------------
log only free kbytes = 5098
device           segment
---------------- ----------
bigdevice        default
bigdevice        system
logdev           logsegment
newdevice        default
newdevice        system
```

The following example creates a segment called new_space on newdevice:

```
sp_addsegment new_space, mydata, newdevice
```

Here is the portion of the sp_helpdb report which lists the segment mapping:

```
device                          segment
------------------------------ ------------------
bigdevice                       default
```

```
bigdevice                      system
logdev                         logsegment
newdevice                      default
newdevice                      new_space
newdevice                      system
```

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 default and system with sp_dropsegment:

```
sp_dropsegment system, mydata, newdevice
sp_dropsegment "default", mydata, newdevice
```

You must include the quotes around “default”; it is a Transact-SQL reserved word.

Here is 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 are not 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 get results like these from sp_helpsegment:

```
create table mytabl (c1 int, c2 datetime)
    on new_space

sp_helpsegment new_space

segment    name                           status
```

```
------- ------------------------------    ------
      3    new_space                            0

device           size        free_pages
---------------- ----------- -----------
newdevice        3.0MB              1523

Objects on segment 'new_space':

table_name        index_name         indid        partition_name
----------------- ------------------ --------     ----------------
mytabl            mytabl                    0     mytabl_400001425

Objects currently bound to segment 'new_space':

table_name      index_name     indid

total_size    total_pages    free_pages  used_pages   reserved_pages
------------ -----------    ----------- -----------   --------------
3.0MB                1536           1523          13                0
```

Segments and clustered indexes

This 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. Check new_space after create index to verify that no objects remain on the segment:

```
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           1523

total_size    total_pages    free_pages  used_pages   reserved_pages
------------ -----------    ----------- -----------   --------------
3.0MB                1536           1530           6                0
```

If you have placed a table on a segment, and you must create a clustered index, use the on *segment_name* clause, or the table migrates to the default segment.

CHAPTER 9

Using the *reorg* Command

Update activity against a table can eventually lead to inefficient utilization of space and reduced performance. The reorg command reorganizes the use of table space and improves performance.

reorg subcommands

The reorg command provides four subcommands for carrying out different types and levels of reorganization:

- reorg forwarded_rows undoes row forwarding.
- reorg reclaim_space reclaims unused space left on a page as a result of deletions and row-shortening updates.
- reorg compact both reclaims space and undoes row forwarding.
- reorg rebuild undoes row forwarding and reclaims unused page space, as does reorg compact. In addition, reorg rebuild:
  - Rewrites all rows to accord with a table's clustered index, if it has one
  - Rewrites space for data and index partitions.
  - Works on individual partitions.

- Writes rows to data pages to accord with any changes made in space management settings through sp_chgattribute
- Drops and re-creates all indexes belonging to the table

The reclaim_space, forwarded_rows, and compact subcommands:

- Minimize interference with other activities by using multiple small transactions of brief duration. Each transaction is limited to eight pages of reorg processing.
- Rewrite space for a single partition.
- Provide resume and time options that allow you to set a time limit on how long a reorg runs and to resume a reorg from the point at which the previous reorg stopped. This allows you to, for example, use a series of partial reorganizations at off-peak times to reorg a large table. For more information, see "resume and time options for reorganizing large tables" on page 212.

The following considerations apply to the rebuild subcommand:

- reorg rebuild holds an exclusive table lock for its entire duration. On a large table this may be a significant amount of time. However, reorg rebuild does everything that dropping and re-creating a clustered index does and takes less time. In addition, reorg rebuild rebuilds the table using all of the table's current space management settings. Dropping and re-creating an index does not use the space management setting for reservepagegap.
- In most cases, reorg rebuild requires additional disk space equal to the size of the table it is rebuilding and its indexes.

The following restrictions hold:

- The table specified in the command, if any, must use either the datarows or datapages locking scheme.
- You must be a System Administrator or the object owner to issue reorg.
- You cannot issue reorg within a transaction.

When to run a *reorg* command

reorg is useful when:

- A large number of forwarded rows causes extra I/O during read operations.
- Inserts and serializable reads are slow because they encounter pages with noncontiguous free space that must be reclaimed.
- Large I/O operations are slow because of low cluster ratios for data and index pages.
- sp_chgattribute was used to change a space management setting (reservepagegap, fillfactor, or exp_row_size) and the change is to be applied to all existing rows and pages in a table, not just to future updates.

Using the *optdiag* utility to assess the need for a *reorg*

To assess the need for running reorg, use statistics from the systabstats table and the optdiag utility. systabstats contains statistics on the utilization of table space, while optdiag generates reports based on statistics in both systabstats and the sysstatistics table.

For information on the systabstats table, see the *Performance and Tuning Guide*. For information about optdiag, see *Utility Programs for UNIX Platforms*.

Space reclamation without the *reorg* command

Several types of activities reclaim or reorganize the use of space in a table on a page-by-page basis:

- Inserts, when an insert encounters a page that would have enough room if it reclaimed unused space
- The update statistics command (for index pages only)
- Re-creating clustered indexes
- The housekeeper garbage collection task, if enable housekeeper GC is set to 1 or more

Each of these has limitations and may be insufficient for use on a large number of pages. For example, inserts may execute more slowly when they need to reclaim space, and may not affect many pages with space that can be reorganized. Space reclamation under the housekeeper garbage collection task compacts unused space, but a single housekeeper garbage collection task that runs at user priority may not reach every page that needs it.

Moving forwarded rows to home pages

If an update makes a row too long to fit on its current page, the row is forwarded to another page. A reference to the row is maintained on its original page, the row's *home* page, and all access to the forwarded row goes through this reference. Thus, it always takes two page accesses to get to a forwarded row. If a scan needs to read a large number of forwarded pages, the I/Os caused by extra page accesses slow performance.

reorg forwarded_rows undoes row forwarding by either moving a forwarded row back to its home page, if there is enough space, or by deleting the row and reinserting it in a new home page. If the table spans partitions, you can specify the partition with the *partition_name* parameter.

You can dipslay statistics on the number of forwarded rows in a table by querying systabstats and using optdiag.

reorg forwarded_rows syntax

The syntax for reorg forwarded_rows is:

```
reorg forwarded_rows table_name partition partition_name
    [with {resume, time = no_of_minutes}]
```

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 212.

reorg forwarded_rows does not apply to indexes, because indexes do not have forwarded rows.

Using *reorg compact* to remove row forwarding

reorg forwarded_rows uses allocation page hints to find forwarded rows. Because it does not have to search an entire table, this command executes quickly, but it may miss some forwarded rows. After running reorg forwarded_rows, you can evaluate its effectiveness by using optdiag and checking "Forwarded row count." If "Forwarded row count" is high, you can then run reorg compact, which goes through a table page by page and undoes all row forwarding.

Reclaiming unused space from deletions and updates

When a task performs a delete operation or an update that shortens row length, the empty space is preserved in case the transaction is rolled back. If a table is subject to frequent deletions and row-shortening updates, unreclaimed space may accumulate to the point that it impairs performance.

reorg reclaim_space reclaims unused space left by deletions and updates. On each page that has space resulting from committed deletion or row-shortening updates, reorg reclaim_space rewrites the remaining rows contiguously, leaving all the unused space at the end of the page. If all rows have been deleted and there are no remaining rows, reorg reclaim_space deallocates the page.

If the table extends over a partition, or several partitions, you can reclaim any available space on the partition by specifying the *partition_name* parameter.

You can display statistics on the number of unreclaimed row deletions in a table from the systabstats table and by using the optdiag utility. There is no direct measure of how much unused space there is as a result of row-shortening updates.

reorg reclaim_space syntax

The syntax for reorg reclaim_space is:

```
reorg reclaim_space table_name [index_name]
    partition partition_name with {resume, time = no_of_minutes}]
```

If you specify only a table name, only the table's data pages are reorganized to reclaim unused space; in other words, indexes are not affected. If you specify an index name, only the pages of the index are reorganized. If you specify a partition, only the part of the table that resides on that partition is affected.

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 212.

Reclaiming unused space and undoing row forwarding

reorg compact combines the functions of reorg reclaim_space and reorg forwarded_rows. Use reorg compact when:

- You do not need to rebuild an entire table (reorg rebuild); however, both row forwarding and unused space from deletes and updates may be affecting performance.
- There are a large number of forwarded rows. See "Using reorg compact to remove row forwarding" on page 205.

reorg compact syntax

The syntax for reorg compact is:

```
reorg compact table_name partition partition_name
    with {resume, time = no_of_minutes}]
```

If you specify a partition, only the part of the table that resides on that partition is affected.

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 212.

Rebuilding a table

Use reorg rebuild when:

- Large I/O is not being selected for queries where it is usually used, and optdiag shows a low cluster ratio for datapages, data rows, or index pages.

- You used sp_chgattribute to change one or more of the exp_row_size, reservepagegap, or fillfactor space management settings and you want the changes to apply not only to future data, but also to existing rows and pages. For information about sp_chgattribute, see the *Reference Manual*.

If a table needs to be rebuilt because of a low cluster ratio, it may also need to have its space management settings changed (see "Changing space management settings before using reorg rebuild" on page 208).

If reorg rebuild finds that the current table is used by another session, it does not wait for this session to end. Instead, it aborts the entire transaction.

reorg rebuild uses a table's current space management settings to rewrite the rows in the table according to the table's clustered index, if it has one. All indexes on the table are dropped and re-created using the current space management values for reservepagegap and fillfactor. After a rebuild, a table has no forwarded rows and no unused space from deletions or updates.

reorg rebuild syntax

The syntax for reorg rebuild is:

reorg rebuild *table_name* [*index_name* [partition *index_partition_name*]

reorg rebuild performs the following when you run it against a table and a partition:

- Takes an exclusive table lock
- Copies data from old to new pages
- Deallocates old data pages
- Locks system tables for updates (including sysindexes, sysobjects, syspartitions, and systabstats)
- Rebuilds clustered and non-clusterd indexes against new data pages
- Commits all open transactions
- Releases locks on system tables

If the table is large and has several indexes, the locks for updating system tables can be held for a long time and can block processes from accessing information in the system tables for the user tables on which you are running reorg. However, because systabstats is already datarow-locked, this system table does not impact this blocking.

reorg rebuild builds the clustered index using the with sorted data option, so the data does not have to be re-sorted during this index build.

Prerequisites for running *reorg rebuild*

Before you run reorg rebuild on a table:

- Set the database option select into/bulkcopy/pllsort to true.
- Determine if your table uses a datapages or datarows locking scheme
- Make sure that additional disk space, equal to the size of the table and its indexes, is available.

To set select into/bulkcopy/pllsort to true:

```
1> use master
2> go
1> sp_dboption pubs2,
    "select into/bulkcopy/pllsort", true
2> go
```

Following a rebuild on a table:

- Dump the database containing the table before you can dump the transaction log.
- Distribution statistics for the table are updated.
- All stored procedures that reference the table are recompiled the next time they are run.

Changing space management settings before using *reorg rebuild*

When reorg rebuild rebuilds a table, it rewrites all table and index rows according to the table's current settings for reservepagegap, fillfactor, and exp_row_size. These properties all affect how quickly inserts cause a table to become fragmented, as measured by a low cluster ratio.

If it appears that a table quickly becomes fragmented and must be rebuilt too frequently, you may need to change the table's space management settings before you run reorg rebuild.

To change the space management settings, use sp_chgattribute (see the *Reference Manual*). For information on space management settings, see the *Performance and Tuning Guide*.

Using the *reorg rebuild* command on indexes

The reorg rebuild command allows you to rebuild individual indexes while the table itself is accessible for read and update activities.

Syntax

The syntax for rebuilding an index is:

```
reorg rebuild table_name [index_name [partition
    index_partition_name] ]
```

Comments

To use reorg rebuild, you must be the table owner or the Database Owner, or have System Administrator privileges.

If you omit the index or partition name, the entire table is rebuilt.

If you specify an index, only that index is rebuilt. Similarly, if you specify an index partition, only that index partition is rebuilt.

Requirements for using reorg rebuild on an index are less stringent than for tables. The following rules apply:

- You do not need to set select into to rebuild an index.
- Rebuilding a table requires space for a complete copy of the table. Rebuilding an index works in small transactions, and deallocates pages once they are copied; therefore, the process needs space only for the pages copied on each transaction.
- You can rebuild the index on a table while transaction level scans (dirty reads) are active.

Limitations

The reorg command applies only to tables using datarows or datapages locking. You cannot run reorg on a table that uses allpages locking.

You cannot run reorg on text that has an indid of 255 in sysindexes.

You cannot run reorg within a transaction.

You can perform a dump tran on a table after rebuilding its index. However, you cannot perform a dump tran if the entire table has been rebuilt.

You can rebuild the index for systabstats, but you cannot run reorg rebuild on the table itself.

Although online index rebuilding is allowed on a placement index, it rebuilds only the index pages. The data pages remain untouched, which means datarows are neither sorted nor rewritten to fresh pages. You can rebuild data pages by dropping a placement index, and then re-creating it.

How indexes are rebuilt with *reorg rebuild index_name partition_name*

Rebuilding a single table or partition index rewrites all index rows to new pages. This improves performance by:

- Improving clustering of the leaf level of the index
- Applying stored values for the fill factor on the index, which can reduce page splits
- Applying any stored value for reservepagegap, which can help reserve pages for future splits

To reduce contention with users whose queries must use the index, reorg rebuild locks a small number of pages at a time. Rebuilding an index is a series of independent transactions, with some independent, nested transactions. Approximately 32 pages are rebuilt in each nested transaction and approximately 256 pages are rebuilt in each outer transaction. Address locks are acquired on the pages being modified and are released at the end of the top action. The pages deallocated in a transaction are not available for reuse until the next transaction begins.

If the reorg rebuild command stops running, the transactions that are already committed are not rolled back. Therefore, the part that has been reorganized is well-clustered with desired space utilization, and the part that has not been reorganized is the same as it was before you ran the command. The index remains logically consistent.

Note Rebuilding the clustered index does not affect the data pages of the table. It only affects the leaf pages and higher index levels. Non-leaf pages above level 1 are not rebuilt.

Space requirements for rebuilding an index

If you do not specify fill_factor or reservepagegap, rebuilding an index requires additional space of approximately 256 pages, or less in the data segment. The amount of log space required is larger than that required to drop the index and re-create it using create index, but it should be only a small fraction of the actual index size. The more additional free space is available, the better the index clustering will be.

Note reorg rebuild may not rebuild those parts of the index that are already well clustered and have the desired space utilization.

Performance characteristics

Index scans traverse faster after you run reorg.

Running reorg against a table can have a negative effect on performance of concurrent queries.

Status messages

Running reorg rebuild *indexname* on a large table may take a long time. Periodic status messages display; starting and ending messages are written to the error log and to the client process executing reorg. In-progress messages display only to the client.

A status reporting interval is calculated as either 10 percent of the pages to be processed or 10,000 pages, whichever is larger. When this number of pages is processed, a status message is printed. Therefore, no more than 10 messages are printed, regardless of the size of the index. Status messages for existing reorg commands are printed more frequently.

resume and *time* options for reorganizing large tables

Use the resume and time options of the reorg command when reorganizing an entire table would take too long and interfere with other database activities. time allows you to run a reorg for a specified length of time. resume allows you to start a reorg at the point in a table where the previous reorg left off. In combination, the two options allow you to reorganize a large table by running a series of partial reorganizations (for example, during off-hours).

resume and time are not available with reorg rebuild.

Syntax for using *resume* and *time* in *reorg* commands

The syntax for resume and time is:

```
reorg forwarded_rows table_name partition partition_name
    [with {resume, time = no_of_minutes}]

reorg reclaim_space table_name [index_name] partition
partition_name
    with {resume, time = no_of_minutes}]

reorg compact table_name partition partition_name
    with {resume, time = no_of_minutes}]
```

The following considerations apply:

- If you specify only the resume option, the reorg begins at the point where the previous reorg stopped and continues to the end of the table.
- If you specify only the time option, the reorg starts at the beginning of the table and continues for the specified number of minutes.
- If you specify both options, the reorg starts at the point where the previous reorg stopped and continues for the specified number of minutes.

Specifying *no_of_minutes* in the *time* option

The *no_of_minutes* argument in the time option refers to elapsed time, not CPU time. For example, to run reorg compact for 30 minutes, beginning where a previous reorg compact finished, enter:

```
reorg compact tablename with resume, time=30
```

If the reorg process goes to sleep during any part of the 30 minutes, it still counts as part of the elapsed time and does not add to the duration of the reorg.

When the amount of time specified has passed, reorg saves statistics about the portion of the table or index that was processed in the systabstats table. This information is used as the restart point for a reorg with the resume option. The restart points for each of the three subcommands that take resume and time options are maintained separately. You cannot, for example, start with reorg reclaim_space and then resume the process using reorg compact.

If you specify *no_of_minutes*, and reorg arrives at the end of a table or an index before the time is up, it returns to the beginning of the object and continues until it reaches its time limit.

Note resume and time allow you to reorganize an entire table or index over multiple runs. However, if there are updates between reorg runs, some pages may be processed twice and some pages may not be processed at all.

CHAPTER 10

Checking Database Consistency

This chapter describes how to check database consistency and perform some kinds of database maintenance using the dbcc commands.

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, checkalloc, checkverify, tablealloc, and indexalloc

- Checking for consistency within and between the system tables in a database with checkcatalog

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.
- 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.
- If 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. For more information, see the *Component Integration Services User's Guide*.

Understanding page and object allocation concepts

When you initialize a database device, the disk init command divides the new space into **allocation units**. The size of the allocation unit depends on the size of the logical pages your server uses (2, 4, 8, or 16K). 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.

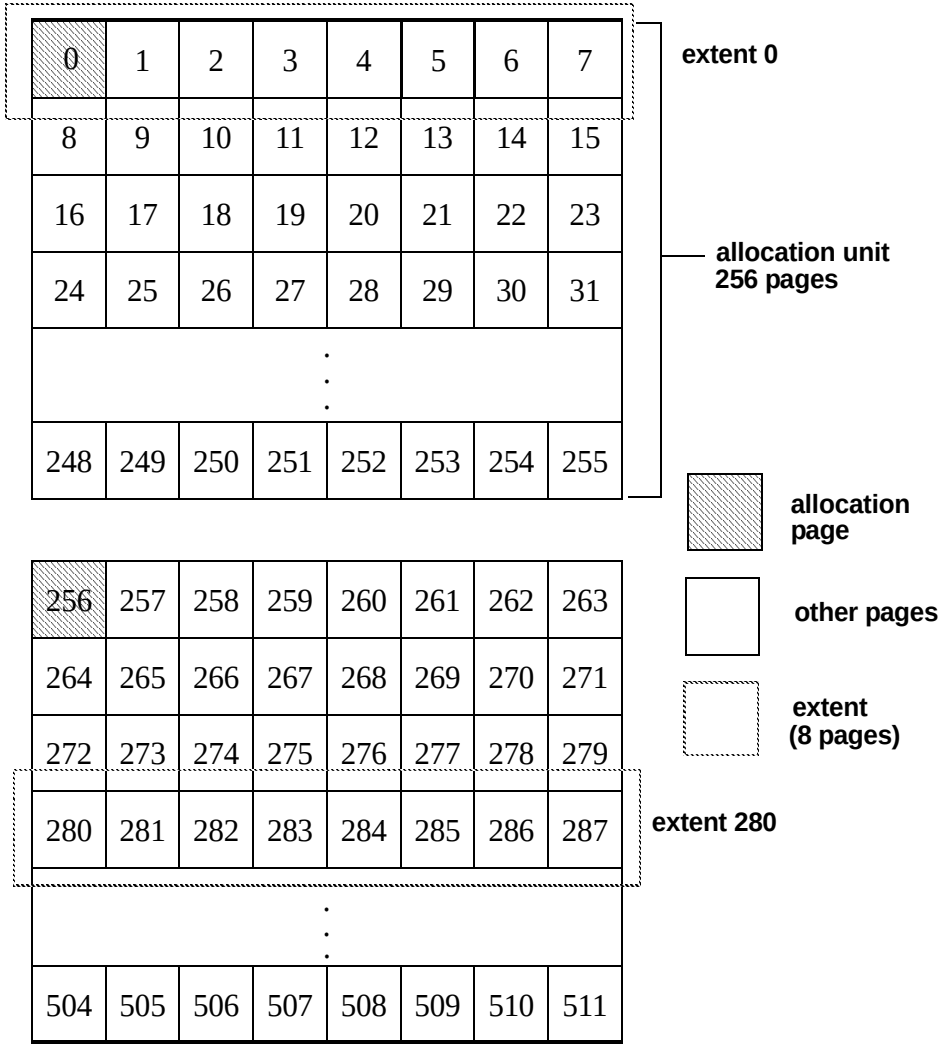
When a table of an indexed partition requires space, Adaptive Server allocates a block of 8 pages to the object. This 8-page block is called an *extent*. Each allocation unit contains 32 extents. The size of the extent also depends on the size of the server logical pages. Adaptive Server uses extents as a unit of space management to allocate and deallocate space as follows:

- When you create a table of an index partition, 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 of an indexed partition, 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 10-1 shows how data pages are set up within extents and allocation units in Adaptive Server databases.

Figure 10-1: 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 information it finds. 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. Each OAM page is the size of one logical page size. For example, on a server using a logical page size of 4K, each OAM page is 4K.

The number of entries per OAM page also depends on the logical page size the server is using. The following table describes the number of OAM entries for each logical page size:

2K logical page size	4K logical page size	8K logical page size	16K logical page size
250	506	1018	2042

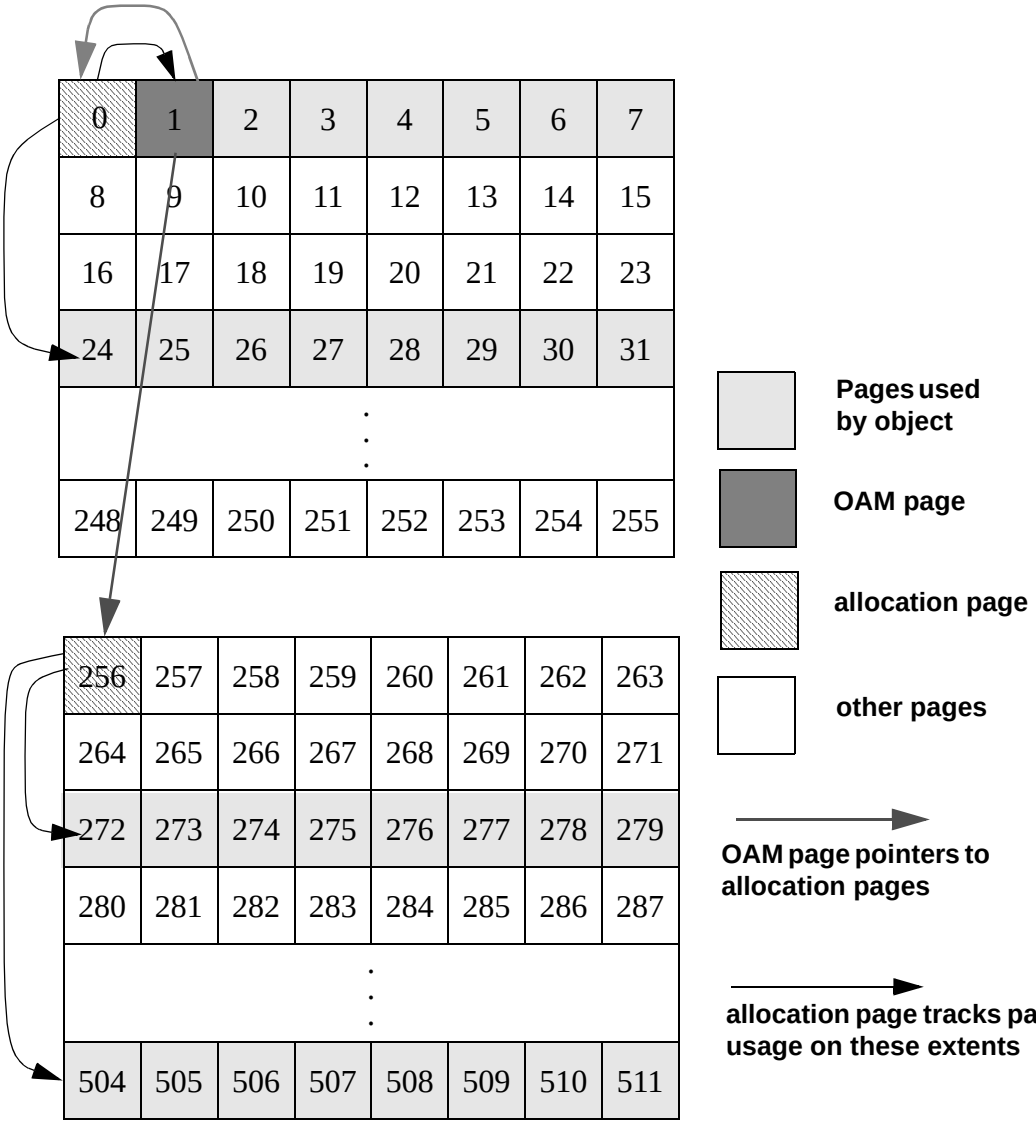
The OAM pages point to the allocation page for each allocation unit where the object uses space. The allocation pages, in turn, track the information about extent and page usage within the allocation unit. In other words, if the titles table is stored on extents 24 and 272, the OAM page for the titles table points to pages 0 and 256.

Figure 10-2 shows an object stored on 4 extents, numbered 0, 24, 272 and 504 for a server that uses 2K logical pages. 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 10-2: OAM page and allocation page pointers

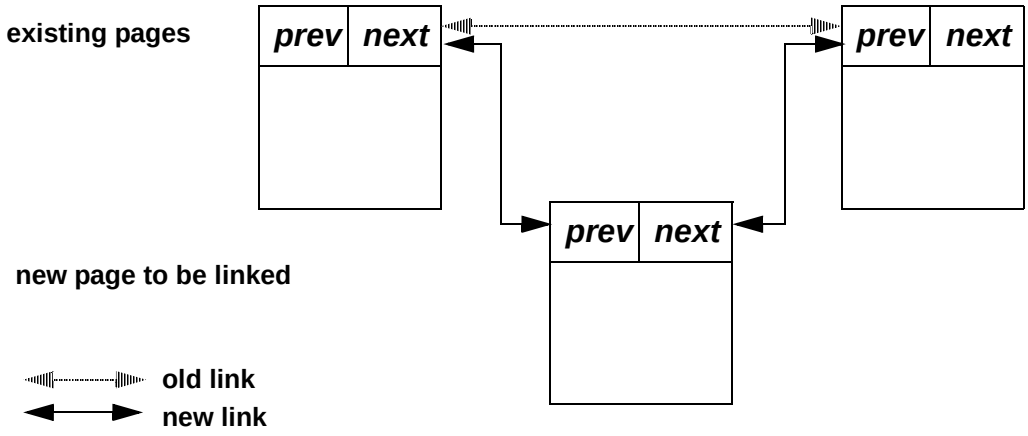


dbcc checkalloc and dbcc tablealloc examine this OAM page information, in addition to checking page linkage, as described in "Understanding page linkage" on page 221.

Understanding page linkage

After a page has been allocated to a table of an indexed partition, that page is linked with other pages used for the same object. Figure 10-3 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 10-3: How a newly allocated page is linked with other pages



What checks can be performed with *dbcc*?

Table 10-1 summarizes the checks performed by the dbcc commands. Table 10-2 on page 237 compares the different dbcc commands.

Table 10-1: Comparison of checks performed by dbcc commands

Checks performed	checkstorage	checktable	checkdb	checkalloc	indexalloc	tablealloc	checkcatalog
Allocation of text valued columns	X						
Index consistency		X	X				
Index sort order		X	X				
OAM page entries	X	X	X		X	X	
Page allocation	X			X	X	X	
Page consistency	X	X	X				
Pointer consistency	X	X	X				
System tables							X
Text column chains	X	X	X	X			
Text valued columns	X	X	X				

Note You can run all dbcc commands except dbrepair and checkdb with the fix option 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
- An OAM page for each partition exists
- Pointer consistency
- Text-valued columns and text-column chains

The following syntax is for dbcc checkstorage, where *dbname* is the name of the **target database** (the database to be checked):

```
dbcc checkstorage [(dbname)]
```

dbcc checkstorage runs checks against the database on disk. If a corruption is only in memory, dbcc checkstorage may not detect the corruption. To ensure consistency between two dbcc checkstorage runs, run checkpoint before running dbcc checkstorage. However, doing so can turn a transient memory corruption into corruption on disk.

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. However, you can run dbcc checkstorage from any database.
- At least two workspaces to use during the check operation. See "dbccdb workspaces" in Chapter 59, "dbccdb Tables" in the *Reference Manual*.
- System and stored procedures to help you prepare your system to use dbcc checkstorage and to generate reports on the data stored in dbccdb.

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.

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 under utilized 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.

   If another session performs drop table concurrently, dbcc checkstorage might fail in the initialization phase. If this happens, run dbcc checkstorage again when the drop table process is finished.

Note See the *Troubleshooting and Error Message Guide* for information about dbcc checkstorage error messages.

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, performance 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 are 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 "dbcc workspaces" in Chapter 59, "dbccdb Tables" in the *Reference Manual*.

To run dbcc checkstorage and one of the system procedures for generating reports with a single command, use sp_dbcc_runcheck.

dbcc checktable

dbcc checktable checks the specified table to see that:

- Index and data pages are linked correctly
- Indexes are sorted properly
- Pointers are consistent
- All indexes and data partitions are correctly linked
- Data rows on each page have entries in the row-offset table; these entries match the locations for the data rows on the page
- Partition statistics for partitioned tables are correct

The syntax for dbcc checktable is:

```
dbcc checktable({table_name | table_id}[, skip_ncindex |
"fix_spacebints" [, "partition_name" | partition_id]])
```

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.

partition_name is the name of the partition you are checking (this may or may not contain the entire table because tables can span multiple partitions), and *partition_id* is the ID of the partition you are checking.

If you specify *partition_name* or *partition_id*, dbcc checktable checks only the table, or parts of the table, residing on this partition; it does not expand its check to other partitions, and has the following restrictions:

- If the table consists of more than one partition, index processing is limited to local indexes.
- If you specify the *partition_name* or *partition_id* parameter, you must also specify either the second parameter (skip_ncindex or fix_spacebits) or null. This example specifies null:

```
dbcc checkalloc(titles, null, 560001995)
```

- If the sort order or character set for a table with columns defined with char or varchar datatypes is incorrect, dbcc checktable does not corrct these values. You must run dbcc checktable on the entire table to correct these errors.
- If an index is marked "read-only" due to a change in the sort order, dbcc checktable does not clear the O_READONLY bit in the sysstat field for the table's sysobjects entry. To clear this status bit, you must run dbcc checktable on the entire table.
- If you run dbcc checktable on syslogs, dbcc checktable does not report space usage (free space versus used space). However, if you do not specify *partition_name* or *partition_id* parameters, dbcc checktable reports the space usage.

When checkstorage returns a fault code of 100035, and checkverify confirms that the spacebit fault is a hard fault, you can use dbcc checktable to fix the reported fault.

The following command checks part of the titles table that resides on the smallsales partition (which contains all book sales less than 5000):

```
dbcc checktable(titles, NULL, "smallsales")
Checking partition 'smallsales' (partition ID 1120003990) of table 'titles'.
The logical page size of this table is 8192 bytes. The total number of data
pages in partition 'smallsales' (partition ID 1120003990) is 1.
Partition 'smallsales' (partition ID 1120003990) has 14 data rows.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

The following is the dbcc checktable syntax, where *table_name* is the name of the table to repair:

dbcc checktable (*table_name*, fix_spacebits)

You can use dbcc checktable 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)

Checking table 'titles' (object ID 576002052):Logical page size is 8192 bytes.
The total number of data pages in partition 'titleidind_576002052' (partition ID
576002052) is 1.
The total number of data pages in this table is 1.
Table has 18 data rows.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

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.

- 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 row_count uses this value to provide fast row estimates in procedures like sp_spaceused.

You can improve dbcc checktable performance by using enhanced page fetching.

dbcc checkindex

dbcc checkindex runs the same checks as dbcc checktable, but only on the specified index instead of the entire table. The synax is:

```
dbcc checkindex({table_name|table_id}, index_id, bottom_up [,
partition_name | partition_id]])
```

partition_name is the name of the partition you are checking and *partition_id* is the ID of the partition you are checking. bottom_up specifies that checkindex checks from the bottom up, starting at the leaf of the index. bottom_up is only applicable for DOL tables. If you specify this option with checkindex or checktable, the index checking is done in a bottom up fashion

dbcc checkdb

dbcc checkdb 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 | fix_spacebits]) ]
```

If you specify the optional skip_ncindex, dbcc checkdb does not check any of the nonclustered indexes on user tables in the database.

If the database extends over a series of partitions, dbcc checkdb performs its checks on each partition.

Checking page allocation

The dbcc commands that you use to check page allocation are:

- dbcc checkalloc
- dbcc indexalloc
- dbcc tablealloc

dbcc checkalloc

dbcc checkalloc ensures that:

- All pages are correctly allocated
- Partition statistics on the allocation pages are correct

- No page is allocated that is not used
- All pages are correctly allocated to individual partitions and that no page that is allocated 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.

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 233.

dbcc checkalloc 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. The INDID values are:

- Tables without clustered indexes have an INDID=0
- For APL tables with clustered indexes, the table data partitions and clustered index partitions are consolidated, with an INDID=1 for the data partitions (or the clustered index partitions).
- For DOL tables with clusetered index, the table data partitions have an INDID=0. The clustered index and nonclustered indexes are numbered consecutively, starting with INDID=2.

Partition and page information is listed after PARTITION ID=*partition_number*.

The following report on pubs2 shows the output for the titleauthor, titles, and stores tables:

```
************************************************************
TABLE: titleauthor OBJID = 544001938
PARTITION ID=544001938 FIRST=904 ROOT=920 SORT=1
Data level: indid 1, partition 544001938. 1 Data pages allocated and 2 Extents
allocated.
Indid : 1, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=928 ROOT=928 SORT=0
```

```
Indid : 2, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=944 ROOT=944 SORT=0
Indid : 3, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 8
************************************************************
TABLE: titles OBJID = 576002052
PARTITION ID=1120003990 FIRST=1282 ROOT=1282 SORT=1
Data level: indid 0, partition 1120003990. 1 Data pages allocated and 1 Extents
allocated.
PARTITION ID=1136004047 FIRST=1289 ROOT=1289 SORT=1
Data level: indid 0, partition 1136004047. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 2
************************************************************
TABLE: stores OBJID = 608002166
PARTITION ID=608002166 FIRST=745 ROOT=745 SORT=0
Data level: indid 0, partition 608002166. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 1
```

dbcc indexalloc

dbcc indexalloc 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, including checking for index partitions. You must specify either an object ID, an object name, or a partition ID, and you must specify an index ID. dbcc checkalloc and dbcc indexalloc include the index IDs in their output.

The syntax for dbcc indexalloc is:

dbcc indexalloc(*object_name* | *object_id* | *partition_id*, *index_id* [, {full | optimized | fast | null} [, fix | nofix]])

To use the fix or nofix option for dbcc indexalloc, you must also specify one of the report options (full, optimized, fast, or null). If you specify a partition ID, only that partition is checked. For details on using the fix and no fix options, see "Correcting allocation errors using the fix | nofix option" on page 233. For details on the reports, see "Generating reports with dbcc tablealloc and dbcc indexalloc" on page 233.

dbcc indexalloc treats unpartitioned indexes as an index with a single partition.

You can run sp_indsuspect to check the consistency of sort order in indexes and dbcc reindex to repair inconsistencies.

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.
- All pages are correctly allocated to the partitions in the specified table and that no page that is allocated is not used.
- No page is used that is not allocated.

The syntax for dbcc tablealloc is:

```
dbcc tablealloc(object_name | object_id | partition_id,
[, {full | optimized | fast | null} [, fix | nofix]])
```

You can specify the table name, the data partition ID, or the table's object ID from the ID column in sysobjects. If you specify a data partition ID, dbcc tablealloc performs its checks on this partition and all local index partitions. If you specify an object name or an object ID, dbcc tablealloc performs its checks on the entire table. If you specify an index partition ID, dbcc tablealloc returns error 15046.

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 233. For details on the reports, see "Generating reports with dbcc tablealloc and dbcc indexalloc" on page 233.

Correcting allocation errors using the *fix | nofix* option

You can use the fix | nofix option with dbcc checkalloc, dbcc tablealloc, and dbcc indexalloc. It specifies 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:

```
sp_dboption dbname, "single user", true
```

You can issue this command only when no one is using the database.

Output from dbcc tablealloc with fix displays allocation errors and 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 you use fix, 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 you do not specify a report type, 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); does not produce an allocation report.

Checking consistency of system tables

dbcc checkcatalog checks for consistency within and between the system tables 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.
- sysindexes is consistent and fixes any errors
- For each row in syspartitions, there exists a matching row in sysegments.
- The last checkpoint in syslogs is valid.

It also lists the segments defined for use by the database and fixes any errors it finds.

The syntax for dbcc checkcatalog is:

```
dbcc checkcatalog [(database_name [, fix])]
```

If you do not specify a database name, dbcc checkcatalog checks the current database.

```
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.
```

Understanding the output from *dbcc* commands

dbcc checkstorage stores the results in the dbccdb database. You can print a variety of reports from this database. For details, see "dbcc checkstorage" on page 223.

The output of most other dbcc commands includes information that identifies the objects being checked and error messages that indicate any problems, the command finds in the object. When you run dbcc tablealloc and dbcc indexalloc with fix, 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(rrtab)

The default report option of OPTIMIZED is used for this run.
The default fix option of FIX   is used for this run.
************************************************************
TABLE: rrtab            OBJID = 416001482
PARTITION ID=432001539 FIRST=2032 ROOT=2040 SORT=1
Data level: indid 1, partition 432001539. 2 Data pages allocated and 2 Extents
allocated.
Indid : 1, partition : 432001539. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=448001596 FIRST=2064 ROOT=2072 SORT=1
Data level: indid 1, partition 448001596. 2 Data pages allocated and 2 Extents
allocated.

Indid : 1, partition : 448001596. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=480001710 FIRST=2080 ROOT=2080 SORT=0
Indid : 2, partition : 480001710. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=496001767 FIRST=2096 ROOT=2096 SORT=0
Indid : 2, partition : 496001767. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=512001824 FIRST=2112 ROOT=2112 SORT=0
Indid : 3, partition : 512001824. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=528001881 FIRST=2128 ROOT=2128 SORT=0
Indid : 3, partition : 528001881. 1 Index pages allocated and 2 Extents
allocated.
```

```
PARTITION ID=544001938 FIRST=680 ROOT=680 SORT=0
Indid : 4, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 18
Alloc page 1792 (# of extent=2 used pages=2 ref pages=2)
Alloc page 1792 (# of extent=2 used pages=3 ref pages=3)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Alloc page 2048 (# of extent=1 used pages=1 ref pages=1)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Alloc page 2048 (# of extent=1 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=3 ref pages=3)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 256 (# of extent=1 used pages=1 ref pages=1)
Alloc page 512 (# of extent=1 used pages=1 ref pages=1)
Total (# of extent=18 used pages=21 ref pages=21) in this database
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role..
```

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 10-2 compares the speed, thoroughness, the level of checking and locking, and performance implications of the dbcc commands. 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 10-1 on page 222 shows which checks are performed by the commands.

Table 10-2: Comparison of the performance of dbcc commands

Command and option	Level	Locking and performance	Speed	Thorough-ness
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 percent faster than without skip_ncindex	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
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 limits for dbcc" on page 256 in the *Performance and Tuning Guide: Optimizer and Abstract Plans* 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 4, "Configuring Data Caches," for information on these commands.

Scheduling database maintenance at your site

There are several factors that determine how often you should run dbcc commands and which ones you need to run.

Database use

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 assure recovery from backups. You can run dbcc checkdb or dbcc checktable less frequently to check index sort order and consistency. This check does not need to be coordinated with any other database maintenance activity. Reserve 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 5, "Distributing Engine Resources," in the *Performance and Tuning Guide: Basics*.

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 is blocked from accessing these tables, even when the command is not checking them.

Backup schedule

The more often you back up your databases and dump your transaction logs, the more data you can recover in case of failure. You 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. You do not have to 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. You can correct problems that occur after loading the dump 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

dbcc checkstorage stores the results in the dbccdb database. You can print a variety of reports from this database. For details, see "dbcc checkstorage" on page 223.

The output of most other dbcc commands includes information that identifies the objects being checked and error messages that indicate any problem, the command finds in the object. When you run dbcc tablealloc and dbcc indexalloc with fix, 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 – 5024 and 100,000 – 100,038. For information on specific errors, see "dbcc_types" on page 1388 of the *Reference Manual*. dbcc checkstorage records two kinds of faults: soft and hard. For information, see "Comparison of soft and hard faults" on page 243.

Errors generated by database consistency problems encountered by dbcc commands other than dbcc checkstorage usually have error numbers from 2500 – 2599 or from 7900 – 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 *Error Messages and 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.

Reporting on aborted checkstorage and checkverify operations

When a checkstorage or checkverify operation aborts, it prints a message that contains the operation's operation id (opid) and the name of the database that was being examined when the operation aborted. An aborted checkverify operation also provides a sequence number in the message, which instructs the user to run sp_dbcc_patch_finishtime, with the provided *dbname*, opid, and (if it was a checkverify operation), the sequence number, *seq*. After executing sp_dbcc_patch_finishtime, you can create fault reports on the aborted operation.

Comparison of soft and hard faults

When dbcc checkstorage finds a fault in the target database, it is recorded in the dbcc_faults table as either a **soft fault** or a **hard fault**. The following sections describe the two kinds of faults. For more information, see "Verifying faults with dbcc checkverify" on page 244.

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 user updates to the database during dbcc checkstorage or when dbcc checkstorage encounters data definition language (DDL) commands. These faults are not repeated when you run the command a second time. You can reclassify soft faults 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 are "persistent" soft faults, and may indicate a corruption. If you execute dbcc checkstorage in single-user mode, the soft faults reported are "persistent" soft faults. You can resolve these faults 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 that cannot be corrected by restarting Adaptive Server. Not all hard faults are equally severe. For example, each of 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.

Verifying faults with *dbcc checkverify*

dbcc checkverify examines the results of the most recent checkstorage operation and reclassifies each soft fault as either a hard fault or an insignificant fault. checkverify acts as a second filter to remove spurious faults from the checkstorage results.

How *dbcc checkverify* works

checkverify reads the recorded faults from dbcc_faults and resolves each soft fault through a procedure similar to that used by the checkstorage operation.

Note checkverify locks the table against concurrent updates, which ensures that the soft faults are reclassified correctly. checkverify does not find errors that have occurred since the last run of checkstorage.

---

checkverify records information in the dbcc_operation_log and dbcc_operation_results tables the same way that checkstorage does. The recorded value of opid is the same as the opid of the last checkstorage operation. checkverify updates the status column in the dbcc_faults table and inserts a row in the dbcc_fault_params table for the faults it processes.

checkverify does not use the scan or text workspaces.

Each fault found by checkstorage is verified by checkverify as one of the following:

- A hard fault classified as such by checkstorage.
- A soft fault reclassified as hard by checkverify because concurrent activity was ruled out as the cause.
- A soft fault confirmed to be soft by checkverify. Some soft faults that appear when there is no concurrent activity in the database do not represent a significant hazard and are not reclassified as hard. A soft fault is not reclassified if it is informational only and not a corruption.
- A soft fault reclassified as insignificant because it can be attributed to concurrent activity or because subsequent activity masked the original inconsistency.

A fault that is assigned code 100011 (text pointer fault) by checkstorage is verified as hard if the text column has a hard fault. If it does not, it is reclassified as soft.

A fault that is assigned code 100016 (page allocated but not linked) by checkstorage is verified as hard if the same fault appears in two successive checkstorage operations. Otherwise, it is reclassified as soft.

When a fault that is assigned code 100035 (spacebits mismatch) by checkstorage is verified as hard, you can repair it by using dbcc checktable.

When checkverify confirms hard faults in your database, follow the same procedures as you did in previous versions of Adaptive Server to correct the faults.

checkverify classifies the following fault codes as soft faults:

- 100020 – check aborted.
- 100025 – row count fault.
- 100028 – page allocation off current segment.

When to use *dbcc checkverify*

You can verify persistent faults by running checkverify anytime after running checkstorage, even after an extended period of hours or days. However, when deciding your schedule, keep in mind that the database state changes over time, and the changes can mask both soft faults and hard faults.

For example, a page that is linked to a table but not allocated is a hard fault. If the table is dropped, the fault is resolved and masked. If the page is allocated to another table, the fault persists but its signature changes. The page now appears to be linked to two different tables. If the page is reallocated to the same table, the fault appears as a corrupt page chain.

Persistent faults that are corrected by a subsequent database change usually do not pose an operational problem. However, detecting and quickly verifying these faults may locate a source of corruption before more serious problems are encountered or before the signature of the original fault changes. For this reason, Sybase recommends that you run checkverify as soon as possible after running dbcc checkstorage.

Note When checkstorage is executed with the target database in single-user mode, there will be no soft faults and no need to execute checkverify.

checkverify runs only one time for each execution of checkstorage. However, if checkverify is interrupted and does not complete, you can run it again. The operation resumes from where it was interrupted.

checkverify may take a long time to complete when processing very large databases. During this time, checkverify does not provide you with any indication of when it will finish. To get a status for the checkverify, use the command_status_reporting command. When the check_status_reporting is on, the checkverify gives progress status reports. The default value is off.

To turn the option on, issue the following command:

```
set command_status_reporting on
```

The results appear as:

```
1> dbcc checkverify (pubs2)
2> go
Verifying faults for’pubs2’.
Verifying faults for table 't1'. The total number of tables to verify is 5. This
is table number 1.
Verifying faults for table 't2'. The total number of tables to verify is 5. This
is table number 2.
Verifying faults for table 't3'. The total number of tables to verify is 5. This
is table number 3.
Verifying faults for table 't4'. The total number of tables to verify is 5. This
is table number 4.
Verifying faults for table 't5'. The total number of tables to verify is 5. This
is table number 5.
DBCC CHECKVERIFY for database ’pubs2’ sequence 4 completed at Apr  9 2003
2:40PM. 72 suspect conditions were resolved as faults, and 11 suspect conditions
were resolved as harmless. 0 objects could not be checked.
```

How to use *dbcc checkverify*

The syntax is:

dbcc checkverify(*dbname*[,*tblname* [,*ignore_exclusions*]])

Where *dbname* is the name of the database to check, *tblname* is the name of the table on which checkverify is to operate, and *ignore_exclusions* is either 0 the default (enables the exclusion list) or 1 (disable the exclusion list).

checkverify operates on the results of the last completed checkstorage operation for the specified database only.

When the checkverify operation is complete, Adaptive Server returns the following message:

```
DBCC checkverify for database name, sequence
n completed at date time. n suspect conditions
resolved as faults and n resolved as innocuous.
n checks were aborted.
```

The following example runs checkverify on the table tab with exclusion list disabled, in the database my_db:

```
dbcc checkverify(my_db, tab)
```

To run dbcc checkverify on table tab, in database my_db, with the exclusion list enabled, enter:

```
dbcc checkverify (my_db, tab, 0)
```

To run dbcc checkverify on table tab, in database my_db, with the exclusion list disabled, enter:

```
dbcc checkverify (my_db, tab, 1)
```

You can run checkverify automatically after running checkstorage by using sp_dbcc_runcheck.

You can exclude tables, faults, and table or fault combinations from checkverify. Use sp_dbcc_exclusions to indicate which items you want excluded from checkverify. sp_dbcc_exclusions is dynamic; that is, checkverify immediately excludes the items you specify in sp_dbcc_exclusions. Once you exclude these objects from checkverify, it does not report on them the next time you run the command. .

You can disable the exclusion list support by entering a value of 1 for the checkverify parameter, *ignore_exclusions*.

Dropping a damaged database

Use dbcc dbrepair dropdb from the master database to drop a damaged 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)
```

The Transact-SQL command drop database does not work on a database that cannot be recovered or used.

Preparing to use *dbcc checkstorage*

Before you can use dbcc checkstorage, you must configure Adaptive Server and set up the dbccdb database. Table 10-3 summarizes the steps and commands in the order you should use them.

Each action is described in detail in the following sections. The examples in this section assume a server that uses 2K logical pages.

Warning! Do not attempt to perform the actions or use the commands in Table 10-3 before you read the information in the referenced section. You must understand the consequences of each action before you make any changes.

Table 10-3: 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 249 "Planning workspace size" on page 251	use master sp_plan_dbccdb
2. If necessary, adjust the number of worker processes that Adaptive Server uses.	"Configuring worker processes" on page 253	sp_configure number of worker processes memory per worker processes
3. *Optional* – create a named cache for dbcc.	"Setting up a named cache for dbcc" on page 255	sp_cacheconfig
4. Configure your buffer pool.	"Configuring an 8-page I/O buffer pool" on page 256	sp_poolconfig

For this action	See	Use this command
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 257	disk init
7. *Optional* – create dbccdb on the data disk device.		create database
8. *Optional* – add disk segments.	"Segments for workspaces" on page 257	use dbccdb
9. Populate the dbccdb database and install dbcc stored procedures.		`isql -Usa -P -i $SYBASE/ASE-12_5/scripts/installdbccdb`

Note dbcc checkstorage runs its checks against the database on disk. If the corruption is only in memory, dbcc may not detect it. To ensure consistency between two sequential dbcc checkstorage commands, first run a checkpoint. However, this can have the side-effect of turning a transient memory corruption into corruption on disk.

Planning resources

Selecting the appropriate device and size for dbccdb is critical to the performance of dbcc checkstorage operations. sp_plan_dbccdb provides 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:

- 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 from a server that uses 2K logical pages 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

(return status = 0)
```

If dbccdb already exists, sp_plan_dbccdb returns:

- 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
(return status = 0)
```

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 *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. To run concurrent dbcc checkstorage operations, to set up additional workspaces.

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—the following 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.

Every non-null text column in the target database 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 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 configure dbccdb to run dbcc checkstorage concurrently on multiple databases. This is possible only when the second and subsequent dbcc checkstorage operations have their own dedicated resources. 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 you run dbcc checkstorage operations serially, the largest scan and text workspaces can be used for all user databases. If you run dbcc checkstorage operations concurrently, set dbccdb to accommodate the largest workspaces that will be used concurrently. Determine the workspace sizes by adding the sizes of the largest databases that will be checked concurrently.

For more information, see "dbccdb workspaces" in Chapter 59, "dbccdb Tables" in the *Reference Manual*.

Automatic workspace expansion

The sp_dbcc_updateconfig. . automatic workspace expansion option indicates whether checkstorage can resize the workspace, if necessary.

At the beginning of a checkstorage run, the size of the workspace is validated. If more space is needed, and automatic workspace expansion is enabled, checkstorage automatically expands the workspace if adequate space is available on the respective segments. If more space is needed, and automatic workspace expansion is not enabled, checkstorage exits and prints an informative message.

A value of 1 enables automatic workspace expansion (the default value). A value of 0 disables enable automatic workspace.

For example, to enable automatic workspace expansion for the database my_db:

```
sp_dbcc_updateconfig my_db, 'enable automatic
workspace expansion', '1'
```

To disable automatic workspace expansion for the database my_db:

```
sp_dbcc_updateconfig my_db, 'enable automatic
workspace expansion', '0'
```

Default scan and text workspaces

Default scan and text workspaces are created when the installdbccdb script is run. The name of the default scan workspace is def$scan$ws and its size is 256K. The default text workspace is named def$text$ws and its size is 128K. These default workspaces are used if you have not configured default workspaces and the target database does not have configured workspace values.

Configuring Adaptive Server for *dbcc checkstorage*

This section provides information on configuring Adaptive Server for dbcc checkstorage.

Configuring worker processes

The following parameters affect dbcc checkstorage:

- max worker processes – set this 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 5, "Setting Configuration Parameters."

max worker processes specifies the maximum number of worker processes used by dbcc checkstorage for each target database, while number of worker processes specifies the total number of worker processes supported by Adaptive Server. Worker processes are not dedicated to running dbcc checkstorage operations.

Set the value for number of worker processes high enough to allow for the number of processes specified by max worker processes. A low number of worker processes reduces the performance and resource consumption of dbcc checkstorage. dbcc checkstorage cannot use more processes than the number of database devices used by the database. 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 does not run.

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 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.

memory per worker process specifies the total memory allocation for worker processes support in Adaptive Server. The default value is adequate for dbcc checkstorage.

Setting up a named cache for *dbcc*

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 which 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 runs concurrently, or you can create one cache that is large enough to fit the total requirements of the concurrent operations. The size 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 buffers (each buffer is one extent) per worker process in the named cache.

For best performance, assign most of the dedicated cache to the buffer pool and do not partition the cache. The recommended cache size is the minimum size for the buffer pool. Add the size of the one page pool to this value.

If you dedicate a cache for dbcc checkstorage, the command does not require more than the minimum one page buffer pool. If the cache is shared, you can improve the performance of dbcc checkstorage by increasing the buffer pool size before running the operation, and reducing the size after the operation is complete. The buffer 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.

Warning! Do not use cache partitions in a cache being used for dbcc checkstorage.

To configure Adaptive Server with a named cache for dbcc checkstorage operations, use sp_cacheconfig and sp_poolconfig. See Chapter 4, “Configuring Data Caches.”

Configuring an 8-page I/O buffer pool

dbcc checkstorage requires a I/O buffer pool of one extent. Use sp_poolconfig to configure the pool size and verify that the pool has been configured properly. The pool size is stored in the dbcc_config table.

The size of the dbcc checkstorage buffer pool is the page size multiplied by 16. The dbcc checkstorage buffer pool requirements for the different page sizes are:

- 2k page server) * (8 extents) = 16k buffer pool
- 4k page server) * (8 extents) = 32k buffer pool
- 8k page server) * (8 extents) = 64k buffer pool
- 16k page server) * (8 extents) = 128k buffer pool

The following example shows how to use sp_poolconfig to set a 16K buffer pool for "master_cache" on a server configured for 2K logical pages. The named cache is 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 *Reference Manual*.

Allocating disk space for *dbccdb*

Additional disk storage is required for the dbccdb database. Because dbcc checkstorage uses dbccdb extensively, place dbccdb on a device that is separate from other database devices.

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 workspaces, you can control the workspace placement and improve the performance of dbcc checkstorage performance. When you dedicate new segments for the exclusive use of workspaces, unmap the devices attached to these segments from the default segment with sp_dropsegment.

Creating the *dbccdb* database

❖ **Creating the *dbccdb* database**

1 Run sp_plan_dbccdb in the master database to obtain recommendations for database size, devices, 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:

```
use master
go
sp_plan_dbccdb pubs2
go
```

The following output appears:

```
Recommended size for dbccdb is 23MB (data = 21MB, log = 2MB).

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 249.

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/",
   size = "4096"
go
disk init
   name = "dbccdb_log",
   physname = "/remote/disks/masters/",
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* – 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 Create the tables for dbccdb and initialize the dbcc_types table:

```
isql -Ujms -P***** -iinstalldbccdb
```

The installdbccdb script checks for the existence of the database before it attempts 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 Create and initialize the scan and text workspaces:

```
use dbccdb|
go|
sp_dbcc_createws dbccdb, scanseg, scan_pubs2, scan, "64K"|
sp_dbccvreatews dbccdb, textseg, text_pubs2, text, "64K"
```

When you have finished installing dbccdb, you must update the dbcc_config table.

Updating the *dbcc_config* table

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

This section describes how to update the values in this table.

Adding default configuration values with *sp_dbcc_updateconfig*

sp_dbcc_updateconfig allows you to provide default configuration values for dbcc configuration parameters. In earlier versions of Adaptive Server, you provided individual configuration values for every database that checkstorage analyzed. Although you can still provide these configuration values for an individual database, you can now set default values that are applicable to all databases that do not have a corresponding configuration value (that is, configuration values cannot conflict).

The syntax for sp_dbcc_updateconfig has not changed; to configure the default values, enter a null *dbname* parameter for sp_dbcc_updateconfig.

For example, after issuing the following, the dbcc configuration parameter max worker processes has a default value of 2:

```
sp_dbcc_updateconfig null, 'max worker processes',
'2'
```

To configure a different value for a specific database, substitute the database name for the null value in the command above:

```
sp_dbcc_updateconfig my_db, 'max worker processes',
'1'
```

Deleting configuration values with *sp_dbcc_updateconfig*

sp_dbcc_updateconfig accepts delete as a value for the *str1* parameter, which allows you to delete configuration values that you have set on databases. The syntax is:

sp_dbcc_updateconfig *dbname*, '*config_parameter*', 'delete'

For example, to delete the default value of the max worker processes configuration parameter, enter:

```
sp_dbcc_updateconfig null, 'max worker processes', 'delete'
```

To delete the value of the max worker processes configuration parameter for the database my_db, enter:

```
sp_dbcc_updateconfig my_db, 'max worker processes', 'delete'
```

Viewing the current configuration values

The sp_dbcc_configeport *defaults* parameter allows you to view the configured default values. The *defaults* parameter is optional, and is ignored if *dbname* is not null. Valid values for the *defaults* parameter are true or false (the default). A value of true indicates that you want to display only the default on the configured values. A value of false indicates that you want to view all configured values.

The syntax for sp_dbcc_configreport is:

```
sp_dbcc_configreport [dbname [, defaults [true | false]]]
```

For example to view only the configured default values, enter true for the *defaults* parameter:

```
sp_dbcc_configreport null, 'true'
```

To view all configured values, do not provide a value for the defaults parameter, or use "false:"

```
sp_dbcc_configreport
```

Or,

```
sp_dbcc_configreport null, 'false'
```

Maintaining *dbccdb*

You occasionally need to perform maintenance tasks on dbccdb.

- Reevaluate and update the configuration using:
  - 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:
  - 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.

- Remove unnecessary workspaces.
- 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 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 may require you to reconfigure 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_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*

Adaptive Server stores data generated by dbcc checkstorage 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 you specify.

When you delete a database, you should also delete from dbccdb all configuration information and dbcc checkstorage results related to that database. Use sp_dbcc_deletedb to delete all database information from dbccdb.

Removing workspaces

You may need to remove unnecessary workspaces. In dbccdb, issue:

```
drop table workspace_name
```

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

A severe corruption in dbccdb may cause dbcc checkstorage to fail. For dbcc checkstorage to locate severe corruptions in dbccdb, you can create an alternate database, dbccalt, which you use only for checking dbccdb. Create dbccalt using the same process that you used to create dbccdb as described in "Preparing to use dbcc checkstorage" on page 248.

If no free devices are available for dbccalt, you can use any device that is not used by the master database or dbccdb.

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 checkstoragez 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 is lost.

Generating reports from *dbccdb*

Several dbcc stored procedures are provided with dbccdb so that you can generate reports from the data in dbccdb.

Reporting 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, see Chapter 10, "dbcc Stored Procedures," in the *Reference Manual*.

Reporting configuration, statistics and fault information

sp_dbcc_fullreport runs these reports in the order shown:

- sp_dbcc_summaryreport – for an example, see "Reporting a summary of dbcc checkstorage operations" on page 264.
- sp_dbcc_configreport – for an example, see "Displaying configuration information for a target database" on page 265.
- sp_dbcc_statisticsreport – for an example, see "Reporting statistics information from dbcc_counter" on page 266.
- sp_dbcc_faultreport short – for an example, see "Reporting faults found in a database object" on page 265.

Displaying 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
```

Comparing 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
```

Reporting 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 for the sybsystemprocs database. The complete report repeats the information for each object in the target database.

```
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:
0x0000374700003788000037460000000 5000648B803EF0001000103FE0080000F
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)
```

Reporting 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
```

Upgrading compiled objects with *dbcc upgrade_object*

Adaptive Server version 11.9.3 introduced the process of upgrading compiled objects based on their source text. Compiled objects are:

- Check constraints
- Defaults
- Rules
- Stored procedures (including extended stored procedures)
- Triggers
- Views

The source text of each compiled object is stored in the syscomments table, unless it has been manually deleted. When you upgrade the server, the existence of the source text in syscomments is verified during that process. However, the compiled objects are not actually upgraded until they are invoked.

For example, if you have a user-defined stored procedure named list_proc, the presence of source text for list_proc is verified when you upgrade to Adaptive Server 12.5.x. The first time list_proc is invoked after the upgrade, Adaptive Server detects that the list_proc compiled object has not been upgraded. Adaptive Server recompiles list_proc, based on the source text in syscomments. The newly compiled object is then executed.

Upgraded objects retain the same object ID and permissions that they used before being upgraded.

Compiled objects for which the source text was hidden using sp_hidetext are upgraded in the same manner as objects for which the source text is not hidden. For information on sp_hidetext, see the *Reference Manual*.

Note If you are upgrading from 32-bit installations to use a 64-bit Adaptive Server, the size of each 64-bit compiled object in the sysprocedures table in each database increases by approximately 55 percent when the object is upgraded. The pre-upgrade process calculates the exact size. Increase your upgraded database size accordingly.

To ensure that compiled objects have been upgraded successfully *before* they are invoked, you can upgrade them manually using the dbcc upgrade_object command. For details, see "Finding compiled object errors before production" on page 269.

Finding compiled object errors before production

Changes made in earlier versions of Adaptive Server may cause compiled objects to work differently in version 12.5.x and later. You can use dbcc upgrade_object to find the following errors and potential problem areas that may require manual changes to achieve the correct behavior:

- Reserved word errors
- Missing, truncated, or corrupted source text
- Quoted identifier errors
- Temporary table references
- select * potential problem areas

After reviewing the errors and potential problem areas, and fixing those that need to be changed, you can use dbcc upgrade_object to upgrade compiled objects manually instead of waiting for the server to upgrade the objects automatically. For details, see "Using dbcc upgrade_object" on page 272.

Reserved word errors

If dbcc upgrade_object finds a reserved word used as an object name in a compiled object, it returns an error, and that object is not upgraded. To fix the error, either manually change the object name or use quotes around the object nam,e and issue the command set quoted identifiers on. Then, drop and re-create the compiled object.

For example, suppose you load a database dump from Adaptive Server 11.5 into Adaptive Server 12.5.x and the dump contains a stored procedure that uses the word "lock." When you run dbcc upgrade_object on that stored procedure, the command returns an error because, although "lock" was not reserved in version 11.5, it became a reserved word in version 11.9.2. With this advance notice, you can change the stored procedure and any related tables before they are used in a production environment.

Missing, truncated, or corrupted source text

If the source text in syscomments was deleted, truncated, or otherwise corrupted, dbcc upgrade_object may report syntax errors. If the source text was not hidden, you can use sp_helptext to verify the completeness of the source text. If truncation or other corruption has occurred, drop and re-create the compiled object.

Quoted identifier errors

dbcc upgrade_object returns a quoted identifier error if:

- The compiled object was created in a pre-11.9.2 version with quoted identifiers active (set quoted identifiers on).
- Quoted identifiers are not active (set quoted identifiers off) in the current session.

To avoid this error, activate quoted identifiers before running dbcc upgrade_object. When quoted identifiers are active, you must use single quotes instead of double quotes around quoted dbcc upgrade_object keywords.

If quoted identifier errors occur, use the set command to activate quoted identifiers, and then run dbcc upgrade_object to upgrade the object.

For compiled objects created in version 11.9.2 or later, the upgrade process automatically activates or deactivates quoted identifiers as appropriate.

Note Quoted identifiers are not the same as literals enclosed in double quotes. The latter do not require you to perform any special action before the upgrade.

Temporary table references

If a compiled object such as a stored procedure or trigger refers to a temporary table (#temp *table_name*) that was created outside the body of the object, the upgrade fails, and dbcc upgrade_object returns an error. To correct this error, create the temporary table exactly as expected by the compiled object, then execute dbcc upgrade_object again. You need not do this if the compiled object is upgraded automatically when it is invoked.

select * potential problem areas

In Adaptive Server version 11.9.3 and later, the results of a select * clause in a stored procedure, trigger, or view that was created in an earlier version of Adaptive Server may be different from what you expect.

For more information about the changes, see the *Reference Manual*.

If dbcc upgrade_object finds a select * clause in the outermost query block of a stored procedure, it returns an error, and does not upgrade the object.

For example, consider the following stored procedures:

```
create procedure myproc as
  select * from employees
go
create procedure yourproc as
  if exists (select * from employees)
    print "Found one!"
go
```

dbcc upgrade_object returns an error on myproc because myproc includes a statement with a select * clause in the outermost query block. This procedure is not upgraded.

dbcc upgrade_object does not return an error on yourproc because the select * clause occurs in a subquery. This procedure is upgraded.

Determining whether *select ** should be changed in views

If dbcc upgrade_object reports the existence of select * in a view, compare the output of syscolumns for the original view to the output of the table, to determine whether columns have been added to or deleted from the table since the view was created.

For example, suppose you have the following statement:

```
create view all_emps as select * from employees
```

Before upgrading the all_emps view, use the following queries to determine the number of columns in the original view and the number of columns in the updated table:

```
select name from syscolumns
  where id = object_id("all_emps")
select name from syscolumns
  where id = object_id("employees")
```

Compare the output of the two queries. If the table contains more columns than the view, and retaining the pre-upgrade results of the select * statement is important, change the select * statement to a select statement with specific column names. If the view was created from multiple tables, check the columns in all tables that comprise the view and rewrite the select statement if necessary.

Warning! Do not execute a select * statement from the view. Doing so upgrades the view and overwrites the information about the original column information in syscolumns.

Another way to determine the difference between the columns in the view and in the new tables is to run sp_help on both the view and the tables that comprise the view.

This comparison works only for views, not for other compiled objects. To determine whether select * statements in other compiled objects need to be revised, review the source text of each compiled object.

Using *dbcc upgrade_object*

Syntax

```
dbcc upgrade_object [ ( dbid | dbname
[, ['database.[owner].]compiled_object_name' |
 'check' | 'default' | 'procedure' | 'rule' |
 'trigger' | 'view'
 [, 'force' ] ] ) ]
```

where:

- *dbid* specifies the database ID. If you do not specify *dbid*, all compiled objects in the current database are upgraded.
- *dbname* specifies the database name. If you do not specify *dbname*, all compiled objects in the current database are upgraded.
- *compiled_object_name* is the name of a specific compiled object you want to upgrade. If you use the fully qualified name, *dbname* and *database* must match, and you must enclose the fully qualified name in quotes. If the database contains more than one compiled object of the same name, use the fully qualified name. Otherwise, all objects with the same name are parsed, and if no errors are found, upgraded.
- check upgrades all check constraints and rules. Referential constraints are not compiled objects and do not require upgrading.
- default upgrades all declarative defaults and the defaults created with the create default command.
- procedure upgrades all stored procedures.
- rule upgrades all rules and check constraints.
- trigger upgrades all triggers.
- view upgrades all views.

The keywords check, default, procedure, rule, trigger, and view specify the classes of compiled objects to be upgraded. When you specify a class, all objects in that class, in the specified database, are upgraded, provided that dbcc upgrade_object finds no errors or potential problem areas.

- force specifies that you want to upgrade the specified object even if it contains a select * clause. Do not use force unless you have confirmed that the select * statement will not return unexpected results. The force option does not upgrade objects that contain reserved words, contain truncated or missing source text, refer to nonexistent temporary tables, or do not match the quoted identifier setting. You must fix these objects before they can be upgraded.

Note If set quoted identifiers is on, use single quotes around the keywords. If set quoted identifiers is off, you can use either double quotes or single quotes.

Examples

```
dbcc upgrade_object
```

Upgrades all compiled objects in the active database.

```
dbcc upgrade_object(listdb, 'procedure')
```

Upgrades all stored procedures in the listdb database. Single quotes are used around procedure because set quoted identifiers is on.

```
dbcc upgrade_object(listdb, "rule")
```

Upgrades all rules and check constraints in the listdb database. Double quotes are used around rule because set quoted identifiers is off.

```
dbcc upgrade_object(listdb, list_proc)
```

Upgrades all stored procedures named list_proc in the listdb database.

```
dbcc upgrade_object(listdb,
  "listdb.jkarrik.list_proc")
```

Upgrades the stored procedure list_proc, which is owned by the login "jkarrik".

```
dbcc upgrade_object(master,
  "listdb.jkarrik.list_proc")
```

Returns an error because the value of *dbname* is master and the value of *database* is listdb. These values must match.

Permissions

Only the Database Owner or a System Administrator can execute dbcc upgrade_object. The Database Owner can upgrade his or her own objects in the database.

Upgraded objects retain the same owner that they had prior to being upgraded.

Increasing the log segment size

You can specify that all compiled objects of a particular class should be upgraded in one execution of dbcc upgrade_object; for example, you can upgrade all triggers by using the trigger keyword. However, even though you use only one dbcc command, the upgrade of each object is recorded in a separate transaction; the old row is deleted from sysprocedures and a new row is written. Therefore, if you run dbcc upgrade_object on a large number of compiled objects, your system may run out of log space. Increase the size of the log segment in the databases in which you plan to run this command, to allow sufficient room to log all the upgrades.

Error reporting

To send all the output from dbcc upgrade_object to the window, a System Administrator can execute dbcc traceon(3604). Sybase recommends that you use this command if you think the output of error messages might overflow the error log.

Using database dumps in upgrades

Upgrading using dump and load

You can load pre-12.5 database dumps and transaction logs and upgrade the databases.

Some issues of which you should be aware:

- Upgrading requires space for copying data and logging changes to the system tables during the upgrade process. If the source database in the dump was nearly full, the upgrade process might fail due to insufficient space. While this is expected to be uncommon, you can use alter database to extend the free space in the event of insufficient-space errors.

- After reloading an older dump, run sp_checkreswords from the new installation on the loaded database to check for reserved words.

Upgrading compiled objects in database dumps

When you load a database dump that was created in an earlier version than the current Adaptive Server, you are not required to perform the pre-upgrade tasks before loading the dump. Therefore, you will not receive any notification if the compiled objects in your database dump are missing their source text. After loading a database dump, run sp_checksource to verify the existence of the source text for all compiled objects in the database. Then, you can allow the compiled objects to be upgraded as they are executed, or you can run dbcc upgrade_object to find potential problems and upgrade objects manually.

For information on using sp_checksource, see the *Reference Manual*.

Determining whether a compiled object has been upgraded

To determine whether a compiled object has been upgraded, do one of the following:

- Look at the sysprocedures.version column. If the object was upgraded, this column will contain the number 12500.
- If you are upgrading to a 64-bit pointer size in the same version, look at the sysprocedures.status column. It will contain a hexadecimal bit setting of 0x2 to indicate that the object uses 64-bit pointers. If the bit is not set, the object is a 32-bit object, which means the object has not been upgraded.

CHAPTER 11

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, make regular and frequent backups of your databases.

This chapter provides information to help you develop a backup and recovery plan. The first part of this chapter provides an overview of Adaptive Server backup and recovery processes. The second part of this chapter discusses the backup and recovery issues that you should address before you begin using your system for production.

Keeping track of database changes

Adaptive Server uses transactions to keep track of all database changes. Transactions are Adaptive Server 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 modifies 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.

Using *delayed_commit* to determine when log records are committed

A relational database is required to ensure a number of transaction properties, including atomicity, consistency, integrity and durability (known as the ACID properties). To ensure this, Adaptive Server adheres to the following rules; it:

- Writes all operations to the transaction log.
- Writes log records before data or index pages are modified.
- Writes log pages to disk when the transaction's commit is issued.
- Notifies the client application of the successful commit only after Adaptive Server has received notification of a successful write to disk from the underlying operating system and IO sub-system.

How does delayed_commit improve performance?

set delayed_commit is a performance option suitable only for certain applications. It increases Adaptive Server's performance for DML operations, (for example, insert, update, delete), but increases the risk of loosing your data during a system failure. Performance gains depend on the application in use.

The types of applications that benefit from set delayed_commit typically include short transactions that are sent rapidly and serially to Adaptive Server. For example, a batch application that issues many insert statements, with each insert being a separate transaction.

You can enable delayed_commit for the session with the set command or the database with sp_dboption. The syntax for delayed_commit is:

```
set delayed_commit on | off | default
```

where on enables the delayed_commit option, off disables it, and default means the database-level setting takes effect.

The syntax for sp_dboption is:

```
sp_dboption database_name, 'delayed commit', [true | false]
```

where true enables delayed_commit at the database level, and false disables the delayed_commit option. Only the DBO can set this parameter.

After you enable set delayed_commit, the client application is notified of a successful commit before the corresponding log records are written to disk. This improves performance because all but the last log page is written to disk, alleviating contention on the last and active log page.

Consider the following before you enable set delayed_commit:

- Issuing shudown with nowait can cause data durability issues unless you issue a checkpoint and it is completed before the server shuts down.
- Enabling set delayed_commit for a session only affects that session. All other sessions' transactions have all their properties enforced, including their ACID properties. This also means other sessions' physical log writes will write the last log page and the log records corresponding to a session with set delayed_commit enabled.
- set delayed_commit is redundant on temporary databases and does not provide a performance improvement.
- Use set delayed_commit only after careful consideration of both your application and operational requirements and your environment. While the risk to data durability may be very low, the options for recovery may be time-consuming if your database is large and your tolerance for missing data is low.

Changes to logging behavior

These are the changes to logging behavior when delayed_commit is enabled.

When a session implicitly or explicitly commits a transaction:

- The user log cache (ULC) is flushed to the transaction log in memory.
- The task issues writes on all non-written log pages except the last (which contains the commit).
- The task notifies the client application of a successful commit without waiting for the IO to complete.

Note This transaction's "last log page" is written:

- By another transaction when it is no longer the "last log page."
- By another, non-delayed transaction when it completes.
- By a checkpoint or the housekeeper buffer wash mechanism.
- By implicit checkpoints causes (for example, shutdown, dump database, dump tran, sp_dboption truncate log on chaeckpoint).

- The task is ready to continue with the next transaction.

Risks of using delayed_commit

When set delayed_commit is enabled, Adaptive Server notifies the client application *before* the actual physical disk write is completed. Because of this, the application can perceive that the transaction is complete whether or not the physical disk write is successful. In the event of a system failure (disk errors, system crash, and so on), transactions that were not written to disk (transactions whose commit records were on the last log page) are not be present after recovery in spite of the application being notified they were committed.

Systems that require tight system interdependencies, such as through a messaging system used through RTDS, further complicate the consequences of using set delayed_commit.

There are two situations where applications can manage the risk:

- The application maintains it's own trace or log, and would, after a system failure, ensure that the database state corresponds to its own trace or log.
- You can restore the database to the state it was in before the application was run. This assumes you took a complete database backup before a batch-job type application is run. In case of failure the database backup is loaded and the batch job is restarted

Enabling *set delayed_commit*

You can enable set delayed_commit for a database or for a session, with the session setting over-ruling the database setting. This means that a session that enables the option has delayed_commit enabled regardless of the database setting.

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. The Adaptive Server automatic **checkpoint** mechanism guarantees that data pages changed by completed transactions are regularly written from the memory cache to the database device. Synchronizing the database and its transaction log shortens the time it takes to recover the database after a system crash.

Setting the recovery interval

Typically, automatic recovery 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 sp_configure with the recovery interval in minutes parameter to specify, the maximum permissible recovery time. Adaptive Server runs automatic checkpoints often enough to recover the database within that period of time:

```
sp_configure "recovery interval in minutes"
```

The default value, 5, allows recovery within 5 minutes per database. To change the recovery interval to 3 minutes, use:

```
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 you create an index on one of its tables.

Automatic checkpoint procedure

Approximately once a minute, the checkpoint task 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 truncate the transaction log when Adaptive Server performs an automatic checkpoint.

To set the trunc log on chkpt database option, which will truncate the transaction log when an automatic checkpoint occurs, 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, design your last-chance threshold procedure to dump the transaction log. For more information about threshold procedures, see Chapter 15, “Managing Free Space with Thresholds.”

Free checkpoints

When Adaptive Server has no user tasks to process, a housekeeper wash task automatically begins writing dirty buffers to disk. If the housekeeper task can flush all active buffer pools in all configured caches, it wakes up the checkpoint task. The checkpoint task determines whether it must perform a checkpoint on the database.

Checkpoints that occur as a result of the housekeeper wash task are known as *free checkpoints*. They do not involve writing many dirty pages to the database device, since the housekeeper wash 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 4, "Using Engines and CPUs," in the *Performance and Tuning Guide: Basics*.

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 recovery mechanisms occur within the recovery interval. (An ordinary shutdown performs a checkpoint.)
- To cause a change in database options to take effect after executing sp_dboption. (After you run sp_dboption, an informational message reminds you to run checkpoint.)

You can use checkpoint to identify the one or more databasess or use an all clause.

```
checkpoint [all | [dbname[, dbname[, dbname.....]]]
```

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.

When you start Adaptive Server, it performs database recovery in this order:

1 Recovers master.

2 Recovers sybsystemprocs.

3 Recovers model.

4 Creates tempdb (by copying model).

5 Recovers sybsystemdb.

6 Recovers sybsecurity.

7 Recovers user databases, in order by sysdatabases.dbid, or according to the order specified by sp_dbrecovery_order. See below for more information about sp_dbrecovery_order.

Users can log in to 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:

```
sp_configure "print recovery information", 1
```

Fast recovery

During a server restart after a planned or unplanned shutdown, or during HA failover, a significant portion of time is spent on database recovery. Faster recovery minimizes database downtime. The goal of fast recovery is to:

- Enhance the performance of database recovery
- Recover multiple databases in parallel by making use of available server resources and tuning them intelligently
- Provide multiple checkpoint tasks at runtime that can run concurrently to minimize the work at recovery time

Adaptive Server start-up sequence

The following is the sequence of events at Adaptive Server start-up:

1 System databases are recovered on engine 0.

2 Adaptive Server accepts user connections.

3 All engines that are configured to be online during start-up are brought online.

4 User databases are recovered in parallel by a "self-tuned" number of recovery tasks using the default data cache tuned for optimal recovery performance.

   For more information about recovery-tuned parameters affecting the default data cache, see "Database recovery" on page 288. For more information about the number of recovery tasks, see "Parallel recovery" on page 287.

During an HA failover, failed over user databases are recovered and brought online in parallel. Fast recovery helps recovery from planned or unplanned shutdowns, unplanned crashes, and HA failover.

Bringing engines online early

Engines are brought online after system databases are recovered, and before user databases. This allows user databases to be recovered in parallel, and makes the engines available for online activities.

Engines are brought online in this fashion during start-up only. In other circumstances, such as failover, engines are already online on the secondary server.

Parallel recovery

With Adaptive Server 12.5.1 and later, during start-up and HA failover, databases are recovered in parallel by multiple recovery tasks. Database recovery is an I/O-intensive process. The time to recover Adaptive Server with parallel recovery depends on the bandwidth of the underlying I/O subsystem. The I/O subsystem should be able to handle Adaptive Server concurrent I/O requests.

With parallel recovery, multiple tasks recover user databases concurrently. The number of recovery tasks is dependent on the configuration parameter max concurrently recovered db. The default value of 0 indicates that Adaptive Server adopts a self-tuning approach in which it does not make any assumptions on the underlying storage architecture. Statistical I/O sampling methods determine the optimal number of recovery tasks depending on the capabilities of the underlying I/O subsystem. An advisory on the optimal number of recovery tasks is provided. If the configuration value is non-zero, Adaptive Server spawns as many tasks as indicated by the configuration parameter.

During parallel recovery, the System Administrator can force serial recovery by reconfiguring the parameter to 1. The active recovery tasks drain out after completing the recovery of the database that is being worked on. The remaining databases are recovered serially.

max concurrently recovered db

max concurrently recovered db determines the degree of parallelism. The minimum value is 1, but you can also use the default value of 0, directing Adaptive Server to use a self-tuning apparoach. The maximum value is the number of engines at startup minus 1. max concurrently recovered db is also limited by the value of the configuration parameter number of open databases.

The default value is 0, which indicates automatic self-tuning by the server to determine the appropriate number of recovery tasks. A value of 1 indicates serial recovery.

Database recovery

Adaptive Server's database recovery includes:

- Log I/O size – Adaptive Server uses the largest buffer pool available in the default data cache for log I/O. If a pool with the largest buffer size is not available, the server creates this pool dynamically, and uses the pool for log I/O. The buffers for this pool come from the default pool. Recovery tunes the size of the large buffer pool for optimal recovery performance. Also, if the large pool is available but the size is not optimal, Adaptive Server dynamically resizes it and the default pool for optimal recovery performance. The buffer pool configurations are restored at the end of recovery.

  For more information on the largest buffer pool, pool sizes, and sp_poolconfig commands, see the *Performance and Tuning Guide: Basics*.

- async prefetch limit – during recovery, the server automatically sets the local async prefetch limit for the pools in the default data cache used by recovery to an optimal value. This overrides any user specifications for the duration of recovery.

  When recovery completes, the original configuration values are restored.

Recovery order

Users can specify the order in which databases are recovered for all or a subset of user databases. You can configure more important databases to be recovered earlier using sp_dbrecovery_order.

Adaptive Server 12.5.1 and later uses parallel recovery tasks to determine the next database to be recovered according to the user-specified order. The remaining databases are recovered in the order of their database IDs. The time to recover a database is dependent on many factors, including the size of the recoverable log. Hence, recovery can complete in an order other than which it started. For applications that must enforce that databases are brought online in the same order as the recovery order, Adaptive Server provides the strict option in sp_dbrecovery_order.

sp_dbrecovery_order has an additional parameter indicating the online ordering.

```
sp_dbrecovery_order [database_name [, rec_order [,
```

```
force [ relax | strict ]]]]
```

- relax – the databases are made as they recover (default).
- strict – the databases are specified by the recovery order.

The default is relax, which means that databases are brought online immediately when recovery has completed.

Parallel checkpoints

A pool of checkpoint tasks works on the list of active databases in parallel. This pool is controlled by the configuration parameter number of checkpoint tasks. Where there is a checkpoint bottleneck, more checkpoint tasks translate to shorter recoverable logs, and recovery has less work to do in case of a crash, thus improving availability.

The default value of number of checkpoint tasks is 1, for serial checkpoints. The number of engines and number of open databases limit the value for this parameter. The absolute maximum value for this parameter is 8. This configuration parameter is dynamic. When the value for this parameter is reduced, checkpoints drain out, and when the value is increased, additional tasks are spawned.

The effectiveness of parallel checkpoints is dependent on the layout of the databases and performance of the underlying I/O subsystem, as checkpoints are I/O-intensive. Tune number of checkpoint tasks depending on the number of active databases and the ability of the I/O subsystem to handle writes.

number of checkpoint tasks

Use the configuration parameter number of checkpoint tasks to configure parallel checkpoints. The maximum value is limited by the value of the configuration parameters number of engines online at startup and number of open databases, with an absolute ceiling of 8.

This parameter is also limited by the value of the configuration parameter number of open databases.

Recovery state

The global variable *@@ recovery_state* determines if Adaptive Server is in recovery. The values that *@@ recovery_state* can have are:

- NOT_IN_RECOVERY – Adaptive Server is not in start-up recovery or in failover recovery. Recovery has been completed and all databases that can be online are brought online.
- RECOVERY_TUNING – Adaptive Server is in recovery (either start-up or failover) and is tuning the optimal number of recovery tasks.
- BOOTIME_RECOVERY – Adaptive Server is in start-up recovery and has completed tuning the optimal number of tasks. Not all databases have been recovered.
- FAILOVER_RECOVERY – Adaptive Server is in recovery during an HA failover and has completed tuning the optimal number of recovery tasks. All databases are not brought online yet.

@@recovery_state can be used by applications to determine when all the databases are recovered and brought online.

Tuning for fast recovery

This section discusses are some guidelines on tuning Adaptive Server to reduce recovery time.

Database layout

- Databases should have logs and data on their own physical devices. The access patterns for log and data are different and should be kept separate.
- Configure the underlying I/O subsystem to handle concurrent I/O requests from multiple databases in Adaptive Server.

Runtime configuration suggestions

- Configure an optimal number of checkpoint tasks using number of checkpoint tasks. Checkpointing is I/O-intensive and should take into account the expected runtime performance and underlying I/O-subsystem performance. Tune number of checkpoint tasks according to the number of active databases and the ability of the I/O subsystem to handle writes. Tune the frequency of checkpoints using the configuration variable recovery interval in minutes.
- Configure an optimal housekeeper wash percentage controlled by housekeeper free write percent, so that during free cycles dirty pages are written out. The default value is usually optimal.
- Configure the order in which user databases should be recovered. The recovery order can be specified for all or a subset of user databases using sp_dbrecovery_order. For such databases, recovery is started in the order specified. Recovery is started in the database ID order, for databases that do not have a recovery order specified. To bring databases online in the order in which they are recovered, use the strict option in sp_dbrecovery_order.
- Ensure that long-running transactions are kept to a minimum. Long-running transactions hold resources and can also cause longer recovery times.
- Shut down the server using polite shutdown to avoid longer recovery times.

Recovery time suggestions

- To facilitate parallel recovery, configure the maximum number of engines to be online at start-up.
- Configure an optimal number of recovery tasks using the configuration parameter max concurrently recovered db. Recovery is I/O-intensive and the value should take into account the response time of the underlying I/O subsystem. This value can be set to 0 (which is the default) for Adaptive Server to determine the optimal number of recovery tasks.
- As recovery uses only the default data cache, cache configuration should ensure that there is a large amount of default data cache for recovery.

- If data space accounting is not essential for a database, set the database option to turn off free space accounting using sp_dboption. This disables threshold actions on the data segment.

User-defined database recovery order

sp_dbrecovery_order allows you to determine the order in which individual user databases recover. This allows you to assign a recovery order in which, for example, critical databases recover before lower-priority databases.

Important features of recovery order are:

- System databases are recovered first, in this order:
  a master
  b sybsystemprocs
  c model
  d tempdb
  e sybsystemdb
  f sybsecurity

  All other databases are considered user databases, and you can specify their recovery order.
- You can use sp_dbrecovery_order to specify the recovery order of user databases and to list the user-defined recovery order of an individual database or of all databases.
- User databases that are not explicitly assigned a recovery order with sp_dbrecovery_order are recovered according to their database ID, after all the databases that have a user-defined recovery order.
- If you do not use sp_dbrecovery_order to assign any databases a recovery order, user databases are recovered in order of database ID.

Using *sp_dbrecovery_order*

To use sp_dbrecovery_order to enter or modify a user-defined recovery order, you must be in the master database and have System Administrator privileges. Any user, in any database, can use sp_dbrecovery_order to the user-defined recovery order of databases.

The syntax for sp_dbrecovery_order is:

```
sp_dbrecovery_order
    [database_name [, rec_order [, force]]]
```

where:

- *database_name* – is the name of the user database to which you want to assign a recovery order.
- *rec_order* – is the order in which the database is to be recovered.

Recovery order must be consecutive, starting with 1. You cannot assign a recovery sequence of 1, 2, 4, with the intention of assigning a recovery order of 3 to another database at a later time.

To insert a database into a user-defined recovery sequence without putting it at the end, enter *rec_order* and specify force. For example, if databases A, B, and C have a user-defined recovery order of 1, 2, 3, and you want to insert the pubs2 database as the second user database to recover, enter:

```
sp_dbrecovery_order pubs2, 2, force
```

This command assigns a recovery order of 3 to database B and a recovery order of 4 to database C.

Changing or deleting the recovery position of a database

To change the position of a database in a user-defined recovery sequence, delete the database from the recovery sequence and then insert it in the position you want it to occupy. If the new position is not at the end of the recovery order, use the force option.

To delete a database from a recovery sequence, specify a recovery order of -1.

For example, to move the pubs2 database from recovery position 2 to recovery position 1, delete the database from the recovery sequence and then reassign it a recovery order as follows:

```
sp_dbrecovery_order pubs2, -1
```

```
sp_dbrecovery_order pubs2, 1, "force"
```

Listing the user-assigned recovery order of databases

To list the recovery order of all databases assigned a recovery order, use:

```
sp_dbrecovery_order
```

This generates output similar to:

```
The following databases have user specified recovery order:
Recovery Order Database Name        Database Id
-------------- -------------------- -----------
    1          dbccdb                     8
    2          pubs2                      5
    3          pubs3                      6
    4          pubtune                    7
The rest of the databases will be recovered in default database id order.
```

To display the recovery order of a specific database, enter the database name:

```
1> sp_dbrecovery_order pubs2
2> go

Database Name Database id Recovery Order
 ------------- ----------- --------------
 pubs2              5           2
```

Fault isolation during recovery

The recovery procedures, known simply as “recovery,” rebuild the server’s databases from the transaction logs. The following situations cause recovery to run:

- Adaptive Server start-up
- Use of the load database command
- Use of the load transaction 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 an index is marked as suspect, the System Administrator can repair this by dropping and re-creating the index.

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. You cannot recover a system database until you have repaired or removed all of its corrupt pages.

Persistence of offline pages

Suspect pages that you have taken offline remain offline when you restart the server. Information about offline pages is stored in master.dbo.sysattributes.

Use the drop database and load database commands to clear entries for suspect pages from master.dbo.sysattributes.

Configuring recovery fault isolation

When Adaptive Server is installed, the default recovery isolation mode is "databases," which marks a database as suspect and takes the entire database offline if it detects any corrupt pages.

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 to set the recovery isolation mode to "page." This mode is in effect the next time that recovery is performed in the database.

The syntax for sp_setsuspect_granularity 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 as suspect, the default behavior is for the database to be accessible for reading and writing with the suspect pages offline and therefore inaccessible. However, if you specify the read_only option to sp_setsuspect_granularity, and recovery marks any pages as suspect, the entire database comes online in read_only mode and cannot be modified. If you prefer the read_only option, but in certain cases you are comfortable allowing users to modify the non-suspect pages, you can make the online portion of the database writable with sp_dboption:

```
sp_dboption pubs2, "read only", false
```

In this case, the suspect pages remain offline until you repair them or force them, as described in "Bringing offline pages online" on page 298.

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 sp_setsuspect_threshold to change 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 sp_setsuspect_granularity and sp_setsuspect_threshold. 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 is "page" and the escalation threshold is 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 sp_listsuspect_db:

```
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.
```

To display detailed information about the individual offline pages, use sp_listsuspect_page. The syntax is:

```
sp_listsuspect_page [dbname]
```

If you do not specify the dbname, the default is 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, and 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 sp_forceonline_db:

```
sp_forceonline_db dbname,
    {"sa_on" | "sa_off" | "all_users"}
```

To make an individual offline page accessible, use sp_forceonline_page:

```
sp_forceonline_page dbname, pgid
    {"sa_on" | "sa_off" | "all_users"}
```

With both of these procedures, you specify the type of access.

- "sa_on" makes 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.
- "sa_off" blocks access to all users, including System Administrators. This reverses a previous sp_forceonline_db or sp_forceonline_page with "sa_on."
- "all_users" brings offline pages online for all users after the pages have been repaired.

  Unlike bringing suspect pages online 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 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.

Index-level fault isolation for data-only-locked tables

When pages of an index for a data-only-locked table are marked as suspect during recovery, the entire index is taken offline. Two system procedures manage offline indexes:

- sp_listsuspect_object
- sp_forceonline_object

In most cases, a System Administrator uses sp_forceonline_object to make a suspect index available only to those with the sa_role. If the index is on a user table, you can repair the suspect index by dropping and re-creating the index.

See the *Reference Manual* for more information about sp_listsuspect_objec and sp_forceonline_object.

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.
- To drop a table or index containing offline pages, you must use a transaction in the master database. Otherwise, the drop fails because it must delete entries for the suspect pages from master.dbo.sysattributes. The following example 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
checkpoint pubs2
go
begin transaction
drop index authors.au_id_ind
commit
go
use master
go
sp_dboption pubs2, "ddl in tran", false
go
checkpoint pubs2
go
```

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

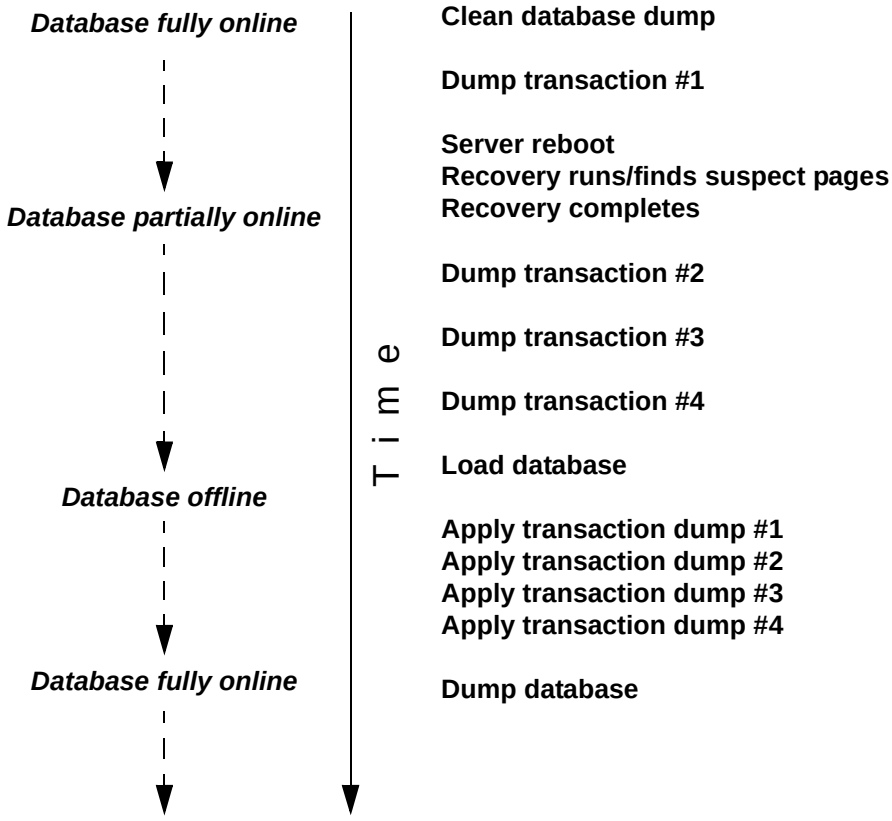
Reloading involves restoring a clean database from backups. When convenient, load the most recent clean database dump, and apply the transaction logs to restore the database.

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 11-1 illustrates the strategy used to reload databases.

Figure 11-1: Reload strategy



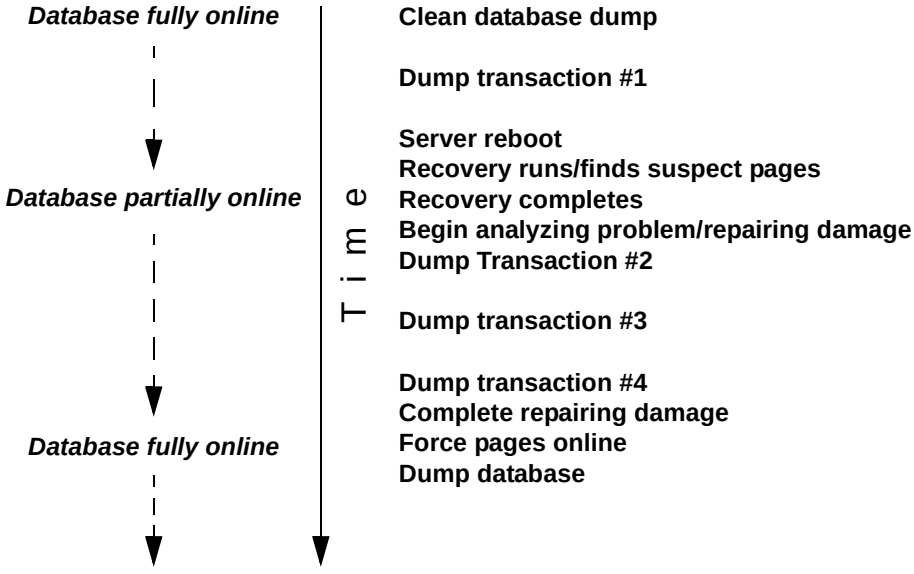
Repair strategy

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, 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.

The repair strategy does not require taking the entire database offline. Figure 11-2 illustrates the strategy used to repair corrupt pages.

Figure 11-2: 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 by restarting Adaptive Server. When recovery completes, 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 this 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 does 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 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 12, "Backing Up and Restoring User Databases," and Chapter 13, "Restoring the System Databases."

Warning! Never use operating system copy commands to copy an operating database device. Running Adaptive Server against a copied device may cause database corruption. You should shutdown Adaptive Server or use the quiesce database command to protect copy operations.

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 10, "Checking Database Consistency," for more information.

Warning! If you dump directly to tape, do not store any other types of files (UNIX backups, tar files, and so on) on that tape. Doing so can invalidate the Sybase dump files. However, if you dump to a UNIX file system, the resulting files can be archived to a tape.

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 makes it convenient to back up databases on a regular basis.

dump database 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 transaction log dump. After 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 back up 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*

If your data device fails and the database is inaccessible, use the with no_truncate option of dump transaction 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 preexisting 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.

The load database command sets the database status to "offline." This means you do not have to use the no chkpt on recovery, dbo use only, and read only options of sp_dboption before you load a database. However, no one can use a 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:

- "Zero" all unused pages, if the database being loaded into is larger than the dumped database.
- Complete recovery, applying transaction log changes to the data.

Depending on the number of unallocated pages or long transactions, this can take a few seconds or many hours for a very large database. Adaptive Server issues messages that it is zeroing pages or has begun recovery. These messages are normally buffered; to see them, issue:

```
set flushmessage on
```

Applying changes to the database: *load transaction*

After 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, using the until_time option of load transaction.

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, change the database status to "online," to make it available to users. A database loaded by load database remains inaccessible until you issue the online database command is issued.

Before you issue this command, be sure you have loaded all required transaction logs.

dump and load databases across platforms

When you perform dump database and load database across platforms with the same endian architecture, user and system data do not require conversions. There are no limitations on operations with the dump and load of a database.

Adaptive Server allows dump and load databases across platforms with different endian architecture. This means you can perform dump database and load database from either a big endian platform to a little endian platform, or from a little endian platform to a big endian platform.

In a big-endian system, the most significant byte of storage, such as integer or long, has the lower address. The reverse is true for a little-endian system.

There is no syntax change with dump or load database. Adaptive Server automatically detects the architecture type of the originating system of the database dump file during a load database, then performs the necessary conversions. Loads from older versions, such as 11.9, 12.0, and 12.5 are supported. The dump and load can be from 32-bit to 64-bit platforms, and vice versa.

Platforms supported:

Big-endian	Sun Solaris	IBM AIX	HP-UX on HPPA, HPIA
Little-endian	Linux IA	Windows	Sun Solaris x86

Dump and load across platforms with the same endian architecture

Stored procedures and other compiled objects are recompiled from the SQL text in *syscomments* at the first execution after the load database for certain combination platforms.

Dump and load across platforms with different endian architecture

Adaptive Server allows a dump and load database between big endian and little endian architectures and vice versa.

Dumping a database

Before you run dump database, for a cross platform dump and load, use the following procedures to move the database to a transactional quiescent status:

1 Verify the database runs cleanly by executing dbcc checkdb and dbcc checkalloc.

2 To prevent concurrent updates from open transactions by other processes during dump database, use sp_dboption to place the database in a single- user mode.

3 Flush statistics to systabstats using sp_flushstats.

4 Wait for 10 to 30 seconds, depending on the database size and activity.

5 Run checkpoint against the database to flush updated pages.

6 Run dump database.

Loading a database

Once you load the database, Adaptive Server automatically identifies the endian type on the dump file and performs all necessary conversions during the load database and online database.

Note After Adaptive Server converts the index rows, the order of index rows may be incorrect. Adaptive Server marks following indexes on user tables as suspect indexes during online database.

- Non-clustered index on APL table.
- Clustered index on DOL table.
- Non-clustered index on DOL table.

Restrictions

- dump transaction and load transaction is not allowed across platforms.
- dump database and load database to or from a remote backupserver are not supported across platforms.
- You cannot load a password-protected dump file across platforms.
- If you perform dump database and load database for a parsed XML object, you must parse the text again after the load database is completed.
- You cannot perform dump database and load database across platforms on Adaptive Servers versions earlier than 11.9.
- Adaptive Server cannot translate embedded data structures stored as binary, varbinary, or image columns.
- load database is not allowed on the master database across platforms.
- Stored procedures and other compiled objects are recompiled from the SQL text in syscomments at the first execution after the load database.

If you do not have permission to recompile from text, then the person who does has to recompile from text using dbcc upgrade_object to upgrade objects.

Note If you migrate login records in *syslogins* system table in the master database from Solaris to Linux, you can use bcp with character format. The login password from the Solaris platform is compatible on Linux without a trace flag from this release. For all other combinations and platforms, login records need to be recreated because the passwords are not compatible.

Performance notes

Due to the design of indexes within a dataserver that provides an optimum search path, index rows are ordered for fast access to the table's data row. Index rows which contain row identifiers (RIDs), are treated as binary to achieve a fast access to the user table.

Within the same architecture platform, the order of index rows remains valid and search order for a selection criteria takes its normal path. However, when index rows are translated across different architectures, this invalidates the order by which optimization was done. This results in an invalid index on user tables when the cross platform dump and load feature is performed.

A database dump from a different architecture, such as big endian to little endian, is loaded, certain indexes are marked as suspect:

- Non-clustered index on APL table.
- Clustered index on DOL table.
- Non-clustered index on DOL table.

To fix indexes on the target system, after load from a different architecture dump, you could use one of two methods:

1 Drop and recreate all of the indexes.

2 Use sp_post_xpload.

Since the data point and information varies from usage on indexes, the schema, user data, number of indexes, index key length, and number of index rows, in general, it requires planning to recreate indexes on large tables as it can be a lengthy process. sp_post_xpload validates indexes, drops invalid indexes, and recreates dropped indexes, in a single command on databases.

Since sp_post_xpload performs many operations it can take longer than drop and recreate indexes. Sybase recommends that you use the drop and recreate indexes on those databases larger that 10G..

Moving a database to another Adaptive Server

You can use dump database and load database 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 ensure that the device allocations on the target Adaptive Server match those on the original. Otherwise, system 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 for recovering a user database from a failed device. See "Examining the space usage" on page 408 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, Adaptive Server attempts 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 user database

You can load dumps into the current version of Adaptive Server from any version of Adaptive Server that is at version 11.9 and later. The loaded database is not upgraded until you issue online database.

The steps for upgrading user databases are the same as for system databases:

1 Use load database to load a database dump of a version 11.9 or later Adaptive Server. load database sets the database status to "offline."

2 Use load transaction to load, *in order*, all transaction logs generated after the last database dump. Load all transaction logs before going to step 3.

3 Use online database to upgrade the database. The online database command upgrades the database because its present state is incompatible with the current version of Adaptive Server. When the upgrade completes, the database status is set to "online," which makes the database available for public use.

4 Make a dump of the upgraded database. A dump database must occur before a dump transaction command is permitted.

For more information about load database, load transaction, and online database, see the *Reference Manual*.

Using the special *dump transaction* options

In certain circumstances, the simple model described above does not apply. Table 11-1 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 11-1. 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, the log may fill completely, causing any further dump transaction commands to fail. Use alter database to allocate additional space to the database.

Table 11-1: 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

When	Use
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

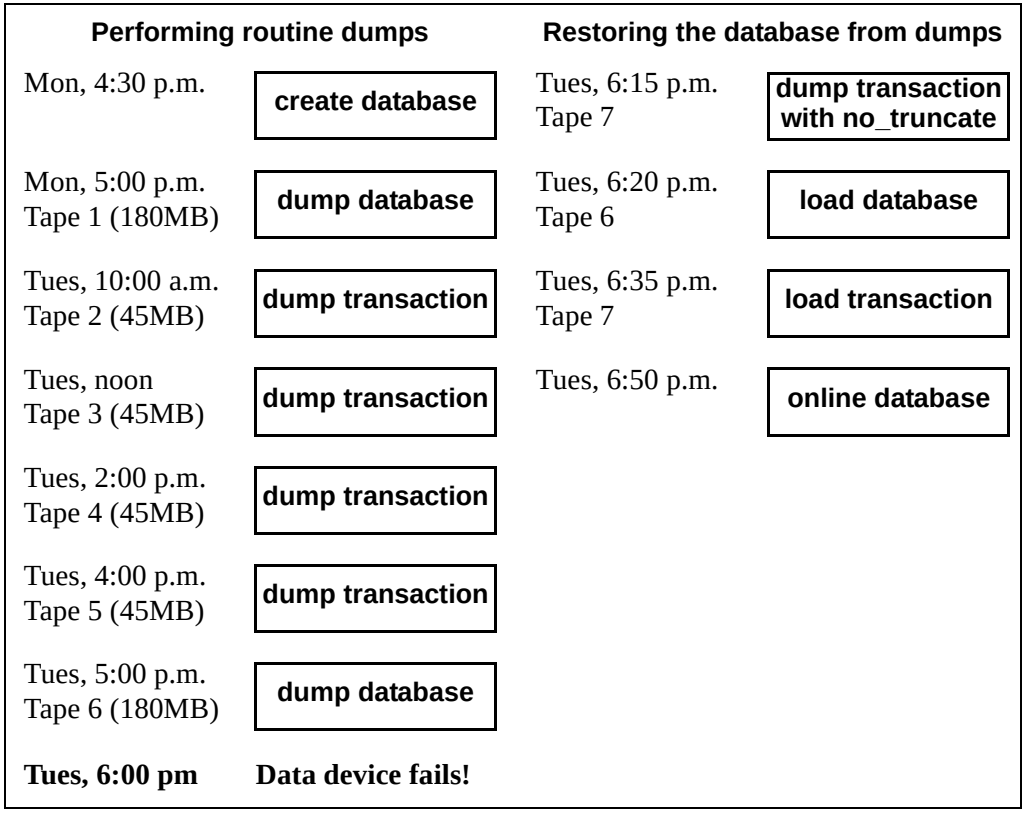
Use the with headeronly option to provide header information for a specified file or for the first file on a tape. Use the with listonly option to return information about all files on a tape. These options do not actually load databases or transaction logs on the tape.

Note These options are mutually exclusive. If you specify both, with listonly prevails.

Restoring a database from backups

Figure 11-3 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 11-3: 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.

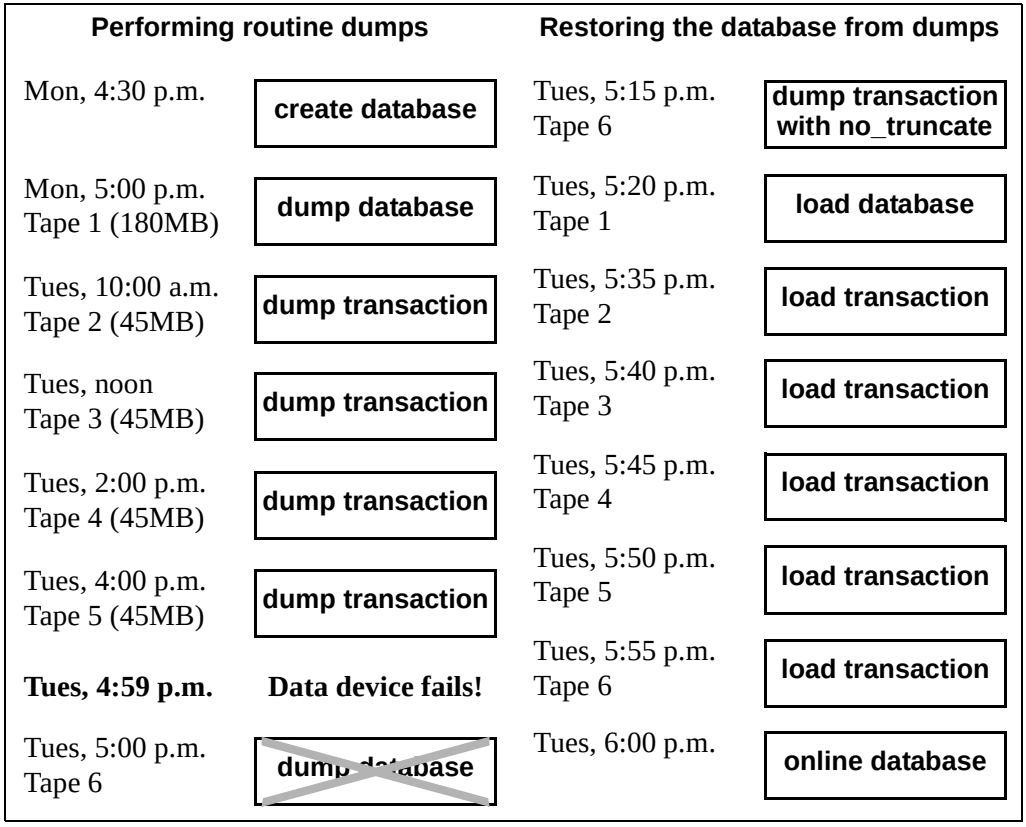
2 Use load database to load the most recent database dump, Tape 6. load database sets the database status to "offline."

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 11-4 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 11-4: 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."

Suspending and resuming updates to databases

quiesce database hold allows you to block updates to one or more databases while you perform a disk unmirroring or external copy of each database device. Because no writes are performed during this time, the external copy (the secondary image) of the database is identical to the primary image. While the database is in the quiescent state, read-only queries to operations on the database are allowed. To resume updates to the database, issue quiesce database release when the external copy operation has completed. You can load the external copy of the database onto a secondary server, ensuring that you have a transactionally consistent copy of your primary image. You can issue quiesce database hold from one isql connection and then log in with another isql connection and issue quiesce database release.

The syntax for quiesce database is:

```
quiesce database tag_name hold database_name
    [, database_name]
[for external dump]
```

or:

```
quiesce database tag_name release
```

where:

- *tag_name* – is a user-defined label for the list of databases to hold or release.
- *database_name* – is the name of a database for which you are suspending updates.

Note *tag_name* must conform to the rules for identifiers. See the *Reference Manual* for a list of these rules. You must use the same *tag_name* for both quiesce database...hold and quiesce database...release.

For example, to suspend updates to the pubs2 database, enter:

```
quiesce database pubs_tag hold pubs2
```

Adaptive Server writes messages similar to the following to the error log:

```
QUIESCE DATABASE command with tag pubs_tag is being executed by process 9.
Process 9 successfully executed QUIESCE DATABASE with HOLD option for tag
pubs_tag. Processes trying to issue IO operation on the quiesced database(s)
will be suspended until user executes Quiesce Database command with RELEASE
option.
```

Any updates to the pubs2 database are delayed until the database is released, at which time the updates complete. To release the pubs2 database, enter:

```
quiesce database pubs_tag release
```

After releasing the database, you can bring up the secondary server with the -q parameter if you used the for external dump clause. Recovery makes the databases transactionally consistent, or you can wait to bring the database online and then apply the transaction log.

Guidelines for using quiesce database

The simplest way to use quiesce database is to make a full copy of an entire installation. This ensures that system mappings are consistent. These mappings are carried to the secondary installation when the system databases that contain them are physically copied as part of quiesce database hold's set of databases. These mappings are fulfilled when all user databases in the source installation are copied as part of the same set. quiesce database allows for eight database names during a single operation. If a source installation has more than eight databases, you can issue multiple instances of quiesce database hold to create multiple concurrent quiescent states for multiple sets of databases.

To create the source installation from scratch, you can use almost identical scripts to create both the primary and secondary installations. The script for the secondary installation might vary in the physical device names passed to the disk init command. This approach requires that updates to system devices on the primary server be reflected by identical changes to the secondary server. For example, if you perform an alter database command on the primary server, you must also perform the same command on the secondary server using identical parameters. This approach requires that the database devices be supported by a volume manager, which can present to both the primary and secondary servers the same physical device names for devices that are physically distinct and separate.

Your site may develop its own procedures for making external copies of database devices. However, Sybase recommends the following:

- Include the master database in quiesce database's list of databases.
- Name devices using identical strings on both the primary and secondary servers.

- Make the environments for the master, model, and sybsystemprocs system databases in the primary and secondary installations identical. In particular, sysusages mappings and database IDs for the copied databases must be identical on the primary and secondary servers, and database IDs for both servers must be reflected identically in sysdatabases.
- Keep the mapping between syslogins.suid and sysusers.suid consistent in the secondary server.
- If the primary server and the secondary server share a copy of master, and if the sysdevices entry for each copied device uses identical strings, the *physname* values in both servers must be physically distinct and separate.
- Make external copies of a database using these restrictions:
  - The copy process can begin only after quiesce database hold has completed.
  - Every device for every database in quiesce database's list of databases must be copied.
  - The external copy must finish before you invoke quiesce database release.
- During the interval that quiesce database provides for the external copy operation, updates are prevented on any disk space belonging to any database in quiesce database's list of databases. This space is defined in sysusages. However, if space on a device is shared between a database in quiesce database's list of databases and a database not in the list, updates to the shared device may occur while the external copy is made. When you are deciding where to locate databases in a system in which you plan to make external copies, you can either:
  - Segregate databases so they do not share devices in an environment where you will use quiesce database, or
  - Plan to copy all the databases on the device (this follows the recommendation above that you make a copy of the entire installation).

- Use quiesce database only when there is little update activity on the databases (preferably during a moment of read-only activity). When you quiesce the database during a quiet time, not only are fewer users inconvenienced, but, depending on the third-party I/O subsystem that is to perform the external copy, there may also be less time spent synchronizing devices involved in the copy operation.
- The mount and unmount commands make it easier to move or copy databases. You can move or copy a database from one Adaptive Server to another without restarting the server. You can move or copy more than one database at a time using the mount and unmount commands.

  You can also use these commands to physically move the devices and then reactivate the databases.

  When you unmount a database, you remove the database and its devices from an Adaptive Server. unmount shuts down the database and drops it from the Adaptive Server; devices are also deactivated and dropped. No changes are made to the database or its pages when unmounted.

Maintaining server roles in a primary and secondary relationship

If your site consists of two Adaptive Servers, one functioning as the primary server, and the other acting as a secondary server that receives external copies of the primary server's databases, you must never mix the roles of these servers. That is, the role each server plays can change (the primary server can become the secondary server and vice-versa), but these roles cannot be fulfilled by the same server simultaneously.

Starting the secondary server with the *-q* option

The dataserver -q option identifies the secondary server. Do not use the -q option to start the primary server. Under the -q option, user databases that were copied during quiesce database for external dump stay offline until:

- You dump the transaction log for a database on the primary server with standby access (that is, dump tran with standby_access) followed by load tran to the copy of this database on the secondary server, and then perform online database for standby access on this database.

- You force the database online for read and write access by issuing online database. However, if you do this, the database recovery writes compensation log records, and you cannot load the transaction log without either loading the database, or making a new copy of the primary devices using quiesce database.

System databases come online regardless of the -q option, and write compensation log records for any transactions that are rolled back.

"in quiesce" database log record value updated

If you start the secondary server using the -q option of dataserver, for each user database marked internally as "in quiesce," Adaptive Server issues a message at start-up stating that the database is `"in quiesce."`

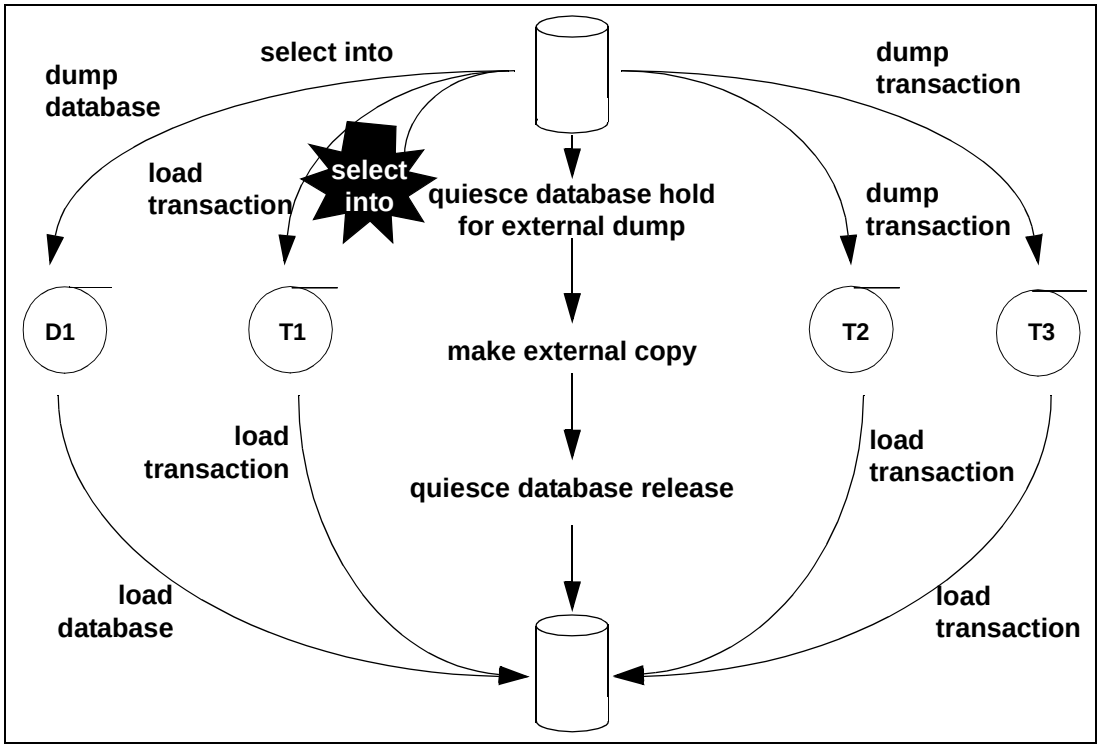
-q recovery for databases copied with quiesce database for external dump acts much like the recovery for load database. Like recovery for load database, it internally records the address of the current last log record, so that a subsequent load transaction can compare this address to the address of the previous current last log record. If these two values do not match, then there has been original activity in the secondary database, and Adaptive Server raises error number 4306.

Updating the dump sequence number

Like dump database, quiesce database updates the dump sequence numbers if there have been unlogged writes. This prevents you from using an earlier database dump or external copy as an improper foundation for a dump sequence.

For example, in the warm standby method that is described in Figure 11-5, archives are produced by dump database (D1), dump transaction (T1), quiesce database, dump transaction (T2), and dump transaction (T3):

Figure 11-5: Warm standby dump sequence



Typically, in an environment with logged updates and no dump tran with truncate_only, you could load D1, T1, T2, and T3 in turn, bypassing any quiesce database hold. This approach is used in a warm standby situation, where succeeding database dumps on the primary server simplify media failure recovery scenarios. On the secondary, or standby server, which is used for decision-support systems, you may prefer continuous incremental applications of load transaction instead of interruptions from external copy operation.

However, if an unlogged operation occurs (for example, a select into, as happens in Figure 11-5) after the dump transaction that produces T1, a subsequent dump transaction to archive is not allowed, and you must either create another dump of the database, or issue quiesce database for external copy and then make a new external copy of the database. Issuing either of these commands updates the dump sequence number and clears the mark that blocks the dump transaction to archive.

Whether or not you use the for external dump clause depends on how you want recovery to treat the quiescent database that would be marked as `in quiesce`.

quiesce database hold

If you issue quiesce database and do not use the for external dump clause, during the external copy operation that creates the secondary set of databases, the secondary server is not running, and recovery under -q does not see any copied database as "in quiesce." It recovers each server in the normal fashion during start-up recovery; it does not recover them as for load database as was previously described. Subsequently, any attempt to perform a load tran to any of these databases is disallowed with error 4306, `"There was activity on database since last load ..."`, or with error 4305, `"Specified file '%.*s' is out of sequence ..."`

Whether or not there been unlogged activity in the primary database, the dump sequence number does not incremented by quiesce database hold, and the unlogged-writes bits are not cleared by quiesce database release.

If you attempt to run a query against a database that is quisced, Adaptive Server issues error message 880:

```
Your query is blocked because it tried to write and
database '%.*s' is in quiesce state. Your query will
proceed after the DBA performs QUIESCE DATABASE
RELEASE
```

The query is run once the database is no longer in a quiescent state.

quiesce database hold for external dump

When you issue quiesce database for external dump, the external copy of the database "remembers" that it was made during a quiescent interval, so that -q recovery can recover it, as happens for load database. quiesce database release clears this information from the primary database. If unlogged writes have prevented dump tran *to archive* on the primary server, dump tran *to archive* is now enabled.

For any database in quiesce database's list, if unlogged writes have occurred since the previous dump database or quiesce database hold for external dump, the dump sequence number is updated by quiesce database hold for external dump, and the unlogged write information is cleared by quiesce database release. The updated sequence number causes load tran to fail if it is applied to a target other than the external copy created under the quiesce database that updated it. This resembles the behavior for dump database of a database with unlogged writes status.

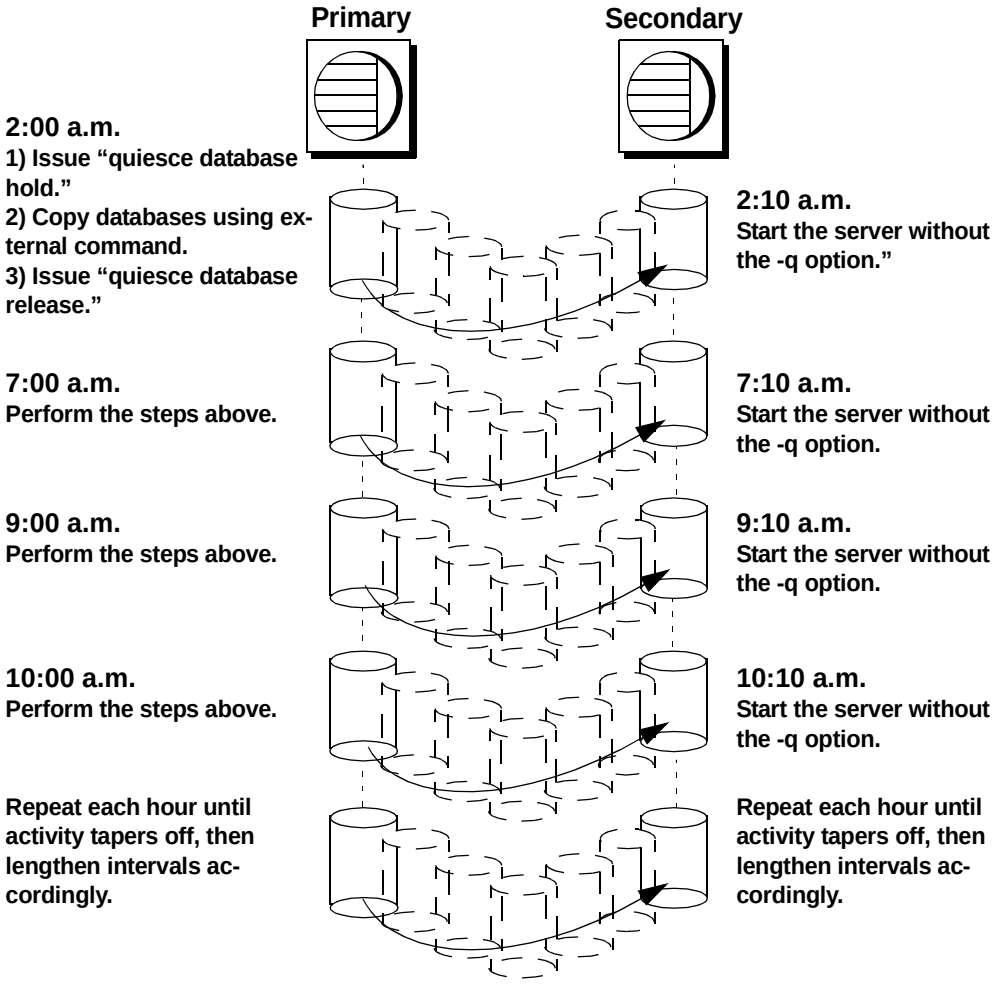
Warning! quiesce database for external dump clears the internal flag that prevents you from performing dump transaction to *archive_device* whether or not you actually make an external copy or perform a database dump. quiesce database has no way of knowing whether or not you have made an external copy. *It is incumbent upon you to perform this duty*. If you use quiesce database hold for external dump to effect a transient write protection rather than to actually perform a copy that serves as the foundation for a new dump sequence, and your application includes occasional unlogged writes, Adaptive Server may allow you to create transaction log dumps that cannot be used. In this situation, dump transaction to *archive_device* initially succeeds, but future load transaction commands may reject these archives because they are out of sequence.

Backing up primary devices with *quiesce database*

Typically, users back up their databases with quiesce database using one of the following methods. Both allow you to off-load decision-support applications from the online transaction processor (OLTP) server during normal operation:

- Iterative refresh of the primary device – copy the primary device to the secondary device at refresh intervals. Quiesce the database before each refresh. A system that provides weekly backups using this system is shown in Figure 11-6:

Figure 11-6: Backup schedule for iterative refresh method

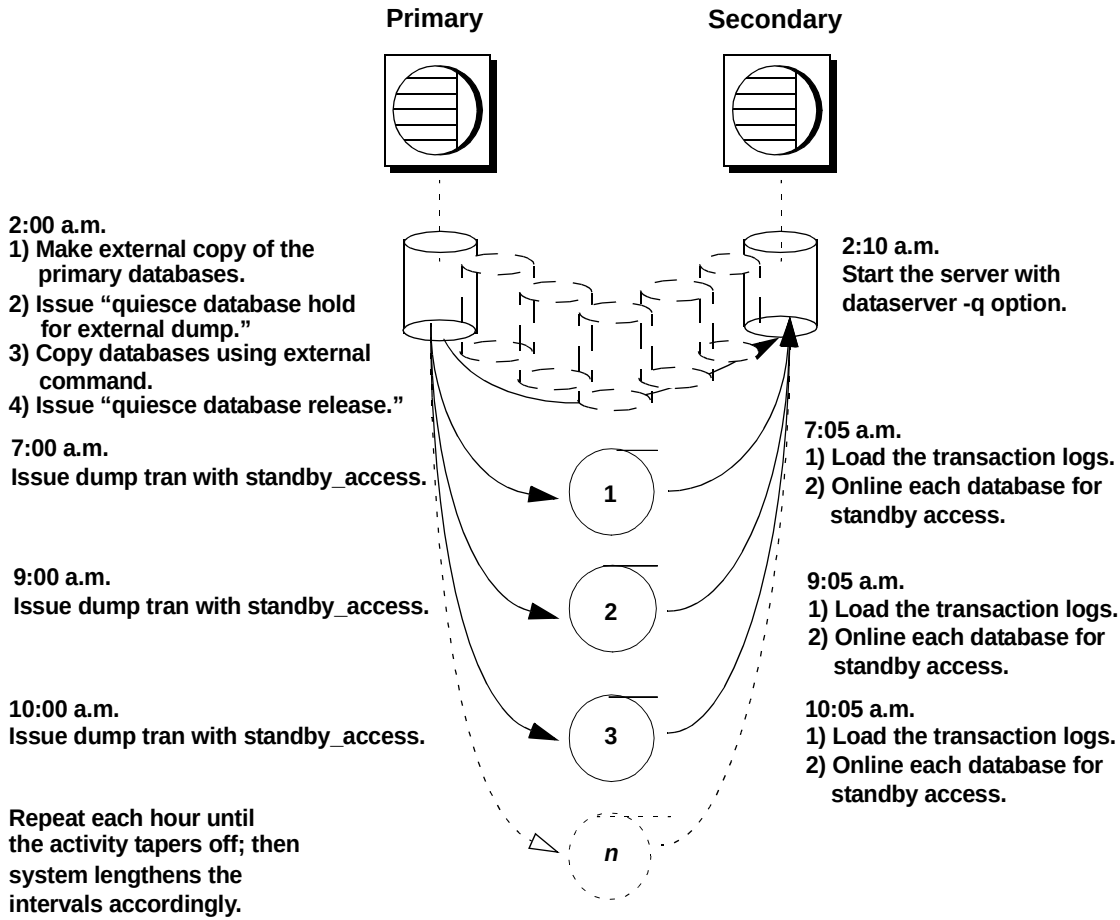


If you are using the iterative refresh method, you do not have to use the `-q` option to restart the secondary server (after a crash or system maintenance). Any incomplete transactions generate compensation log records, and the affected databases come online in the regular fashion.

- Warm standby method – allows full concurrency for the OLTP server because it does not block writes.

After you make an external (secondary) copy of the primary database devices using the for external dump clause, refresh the secondary databases with periodic applications of the transaction logs with dumps from the primary server. For this method, quiesce the databases once to make an external copy of the set of databases and then refresh each periodically using a dump tran with standby_access. A system that uses a daily update of the primary device and then hourly backups of the transaction log is shown in Figure 11-7.

Figure 11-7: Backup schedule for warm standby method



Recovery of databases for warm standby method

If you are using the warm standby method, Adaptive Server must know whether it is starting the primary or the secondary server. Use the -q option of the dataserver command to specify that you are starting the secondary server. If you do not start the server with the -q option:

- The databases are recovered normally, not as they would be for load database.
- Any uncommitted transactions at the time you issue quiesce database are rolled back.

See "Starting the secondary server with the -q option" on page 319 for more information.

The recovery sequence proceeds differently, depending on whether the database is marked `in quiesce`.

Recovery of databases that are not marked "in quiesce"

Under the -q option, if a database is not marked `in quiesce`, it is recovered as it would be in the primary server. That is, if the database is not currently in a load sequence from previous operations, it is fully recovered and brought online. If there are incomplete transactions, they are rolled back and compensation log records are written during recovery.

Recovery of databases that are marked as "in quiesce"

- User databases – user databases that are marked `in quiesce` recover in the same manner as databases recovering during load database. This enables load tran to detect any activity that has occurred in the primary database since the server was brought down. After you start the secondary server with the -q option, the recovery process encounters the `in quiesce` mark. Adaptive Server issues a message stating that the database is in a load sequence and is being left offline. If you are using the warm standby method, do not bring the database online for its decision-support system role until you have loaded the first transaction dump produced by a dump tran with standby_access. Then use online database for standby_access.
- System databases – system databases come fully online immediately. The `in quiesce` mark is erased and ignored.

Making archived copies during the quiescent state

quiesce database hold for external dump signifies your intent to make external copies of your databases during the quiescent state. Because these external copies are made after you issue quiesce database hold, the database is transactionally consistent because you are assured that no writes occurred during the interval between the quiesce database hold and the quiesce database release, and recovery can be run in the same manner as start-up recovery. This process is described in Figure 11-5 on page 321.

If the environment does not have unlogged updates and does not include a dump tran with truncate_only, you might load D1, T1, T2, and T3 in turn, bypassing any quiesce database...hold commands. However, if an unlogged operation (such as a select into shown in Figure 11-5) occurs after the dump transaction that produces T1, dump transaction to archive is no longer allowed.

Using the quiesce database hold for external dump clause addresses this problem by clearing the status bits that prevent the next dump transaction to archive and changing the sequence number of the external copy to create a foundation for a load sequence. However, if there have been no unlogged writes, the sequence number is not incremented.

With or without the for external dump clause, you can make external copies of the databases. However, to apply subsequent transaction dumps from the primary to the secondary servers, you must include the for external dump clause, as shown:

```
quiesce database tag_name hold db_name [, db_name]
... [for external dump]
```

For example:

```
quiesce database pubs_tag hold pubs2 for external
dump
```

Assuming the database mappings have not changed since the primary instance of the database was initiated, the steps for making an external dump are for a single database include:

1 Issue the quiesce database hold for external dump command:

```
quiesce database pubs_tag hold pubs2 for external
dump
```

2 Make an external copy of the database using the method appropriate for your site.

3 Issue quiesce database *tag_name* release:

```
quiesce database pubs_tag release
```

Warning! Clearing the status bits and updating the sequence number enables you to perform a dump transaction whether or not you actually make an external copy after you issue quiesce database. Adaptive Server has no way of knowing whether or not you have made an external copy during the time between quiesce database... hold for external dump and quiesce database... release. If you use the quiesce database hold for external dump command to effect a transient write protection rather than to actually perform a copy that can serve as the foundation for a new dump sequence, and your application includes occasional unlogged writes, Adaptive Server allows you to create transaction log dumps that cannot be used. dump transaction to *archive_device* succeeds, but load transaction rejects these archives as being out of sequence.

Using *mount* and *unmount* commands

The mount and unmount commands make it easier to move or copy databases. You can move or copy a database from one Adaptive Server to another without restarting the server. You can move or copy more than one database at a time using the mount and unmount commands.

You can also use these commands to physically move the devices and then reactivate the databases.

Warning! Direct mapping to a login name is not maintained within a database in Adaptive Server. This means that, for every login allowed access to a database on the original Adaptive Server, a corresponding login for the same suid must exist at the destination Adaptive Server.

For permissions and protections to remain unchanged, the login maps at the secondary Adaptive Server must be identical to the files on the first Adaptive Server.

Manifest file

The manifest file is the binary file which describes the databases that are present on a set of database devices. It can be created only if the set of databases that occupy those devices are isolated, and self-contained on those devices.

Since the manifest is a binary file, operations that can perform character translations of the file contents (such as ftp) corrupt the file unless executed in binary mode.

Performance considerations

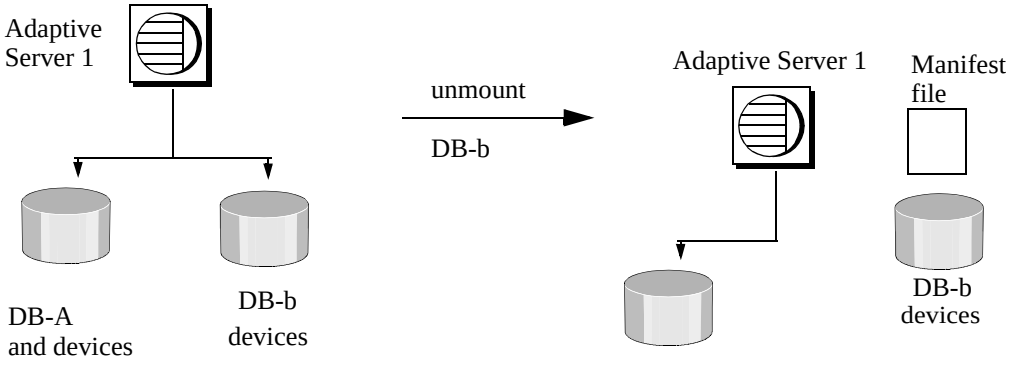
A change in dbid results in all procedures being marked for recompiling in the database. This affects the time taken to recover the database at the destination and the first execution of the procedure.

System restrictions

- You cannot unmount system databases. However, you can unmount sybsystemprocs.
- You cannot unmount proxy databases cannot be unmounted.
- mount and unmount database commands are not allowed in a transaction.
- mount database is not allowed in an HA-configured server.

Unmounting the database

Figure 11-8: unmount database command



Warning! The unmount command removes a database and all its information from the Adaptive Server. Use the unmount command with extreme caution.

When you unmount a database, you remove the database and its devices from an Adaptive Server. The unmount command shuts down the database and drops it from the Adaptive Server. The devices are also deactivated and dropped. The database and its pages are not altered when they are unmounted. The database pages are still present on the OS devices.

```
unmount database <dbname list> to <manifest_file>
```

Use quiesce database to create the *manifest* file, then execute unmount. For example:

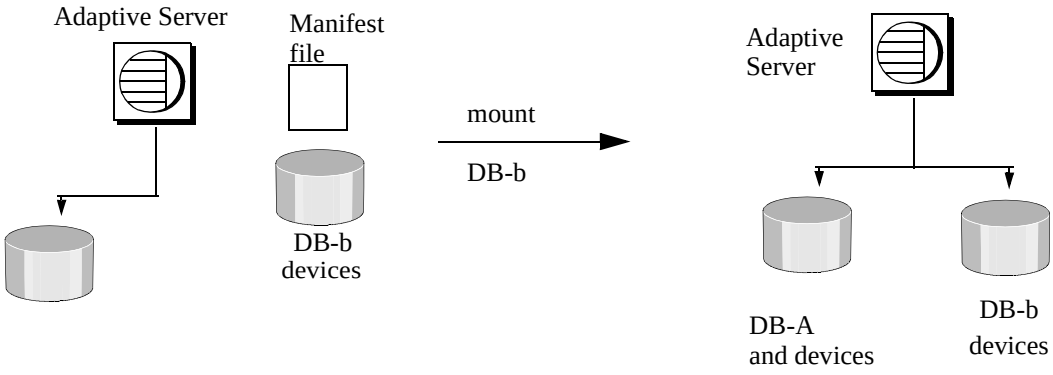
```
1> unmount database pubs2 to
"/work2/Devices/Mpubs2_file"
2> go
```

If you now try to use the pubs2 database, you see this error:

```
Attempt to locate entry in sysdatabases for database
'pubs2' by name failed - no entry found under that
name. Make sure that name is entered properly.
```

Mounting a database

Figure 11-9: mount command



Use the mount command to add the information for devices and other attributes for the database to the destination or secondary Adaptive Server. The mount command decodes the information in the manifest file and makes the set of databases available online. All the required supporting activities are done, including adding database devices if not present and activating them, creating the catalog entries for the new databases, recovering them, and putting them online. When you mount databases onto an Adaptive Server:

- You cannot mount a subset of the databases described in the manifest. All the databases and devices present in the manifest must be mounted together.
- The databases being mounted must have the same page size as the previous Adaptive Server.
- There must be enough devices configured on the secondary Adaptive Server for the successful addition of all the devices belonging to the mounted databases.
- The configuration parameter number of devices must be set appropriately.
- Database names and devices with the same names as the mounted database must not already exist.
- Adaptive Server must have the same version as the mounted database.

- The mounted database must be from the same platform as the Adaptive Server.

Note Remember that if you use the unmount command, the database is removed from the original Adaptive Server with its information on attributes, device names, and so on.

Syntax:

mount all from <*manifest_file*>

For example:

```
1> mount database all from
"/work2/Devices/Mpubs_file"
2> go

Redo pass of recovery has processed 1 committed and
0 aborted transactions.
MOUNT DATABASE: Completed recovery of mounted
database 'pubs2'
```

Once the mount is completed, the database is still offline. Use the online database command to bring the database online. You do not have to restart the server.

Renaming devices

The manifest file contains the device paths as known to the Adaptive Server that created the manifest file. If the mounting Adaptive Server accesses the devices with a different path, you can specify the new path to the mount command.

1 Use the mount command with listonly to get the old path:

```
1> mount database all from "/work2/Mpubs_file"
with listonly
2> go

"/work2/Devices/pubsdat.dat" = "pubs2dat"
```

2 If the new path for the device *pubs2dat* is /work2/Devices/pubsdevice.dat, specify the new device in the mount command:

```
mount database all from "/work2/Mpubs_file" using
"work2/datadevices/pubsdevice.dat" = "pubs2dat"
```

quiesce database

Use the quiesce database hold command with the manifest clause to hold the database and create a manifest file.

Note Since the manifest file is binary, operations that perform character translations of the file contents (such as ftp) corrupt the file unless performed in binary mode.

---

quiesce database < *tag_name* > hold < *dbname list* >
[for external dump] [to <*manifest_file*> [with override]]

For example, place the database in a hold status and build the manifest file:

```
quiesce database pubs2_tag hold pubs2 for external
dump to "/work2/Devices/Mpubs_file"
```

This stops transactions against the database and creates the manifest file. To verify the devices within the manifest file, execute mount with listonly:

```
mount database all from "/work2/Devices/Mpubs_file"
with listonly

"/work2/Devices/pubsdat.dat" = "pubs2dat"
```

Execute the quiesce database hold, then copy the database devices.

You cannot create a manifest file if the set of databases that are quiesced contain references to databases outside of the set. You may use the override option to bypass this restriction.

Creating a mountable copy of a database

1 Use the quiesce database command with the manifest clause and quiesce the database. This command creates a manifest file describing the database.

2 Use the mount command with listonly to display the list of devices to be copied.

3 Use external copy utilities, such as cp, dd, split mirror, and so on, to copy the database devices to another Adaptive Server.

The copy of the devices and the manifest file is a mountable copy of the database.

Moving databases from one Adaptive Server to another

1. Use the unmount command to unmount the database from the first Adaptive Server. The command creates a manifest file describing the database.
2. Make the database devices available to the second Adaptive Server, if not already available. This may require the help of your OS administrator if the second Adaptive Server is on another machine.
3. Execute the mount command on the secondary Adaptive Server with the manifest file created in step 1.

Designating responsibility for backups

Many organizations have an operator who performs 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.

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 Backup Server on a remote computer and another on the local computer.

Note Backup Server cannot dump to multidisk volumes.

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

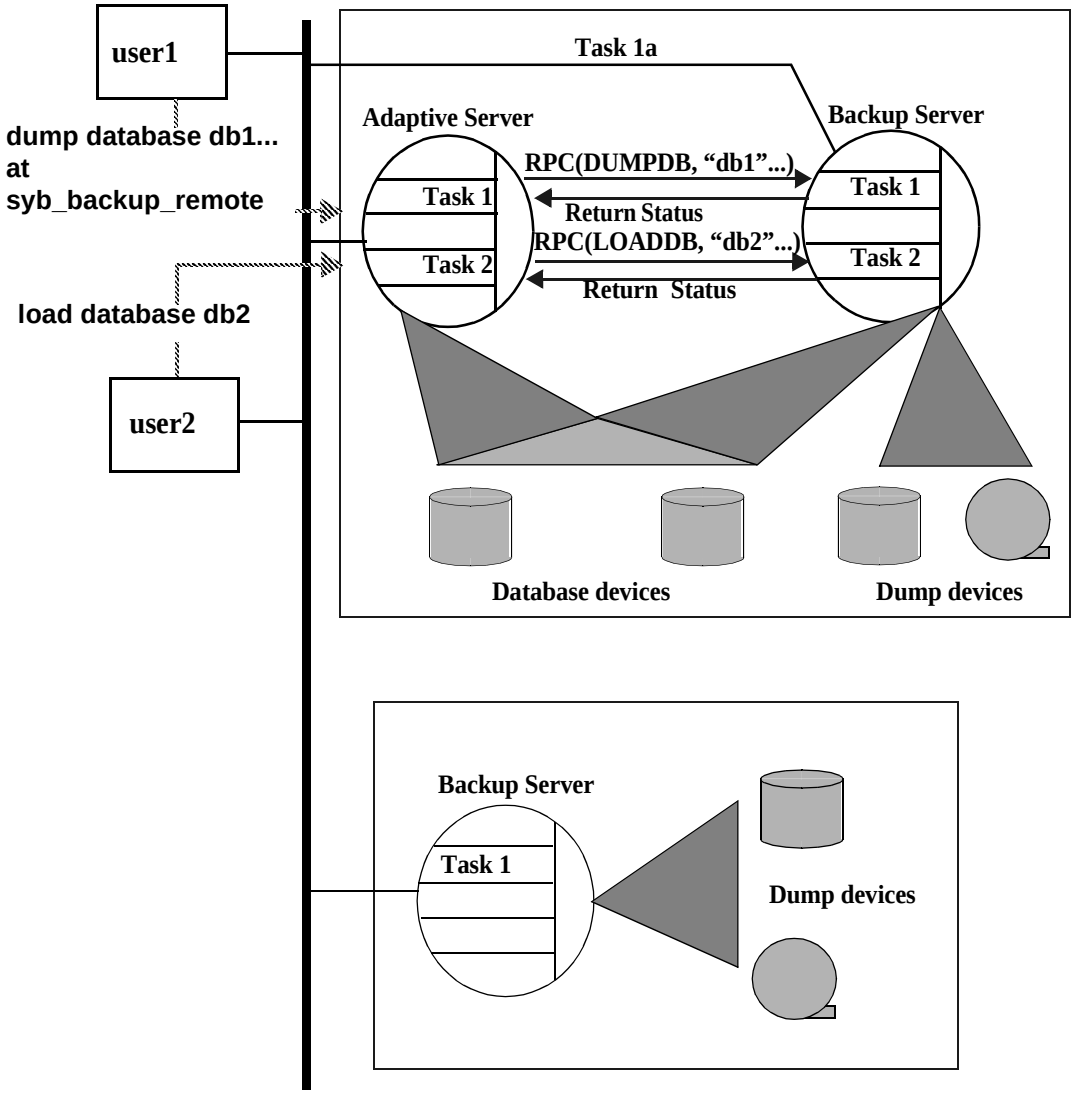
Figure 11-10 shows two users performing backup activities simultaneously on two databases:

- 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 11-10: 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 339 for more information.

Mounting a new volume

Note Backup Server cannot dump to multidisk volumes.

During the backup and restore process, 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 sp_volchanged 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 11-2).

Table 11-2: 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

Sequence	Operator using isql	Adaptive Server	Backup Server
8	Receives message that dump is complete Removes and labels tapes	Receives message that dump is complete 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.

Use shutdown to shut down a Backup Server. See Chapter 11, "Diagnosing System Problems," and the *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 when dumps and loads are taking place. To disable remote access, use:

```
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
```

allow remote access is dynamic and does not require a restart of Adaptive Server to take effect. Only a System Security Officer can set allow remote access.

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 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. The default value of tape retention in days is 0. Which 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 dump database and dump transaction 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.

You can copy dumps to a file or disk to tape for offline storage, but these tapes must be copied back to an online file before they can be read by Adaptive Server. Backup Server cannot directly read a dump that is made to a disk file and then copied to tape.

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, and whenever possible, you must modify scripts and procedures that refer to actual device names each time you replace a backup device. If you use 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.

Note Make sure that the device driver options you include with the dump command are accurate. Backup Server does not verify any device driver options you include during a dump command. For example, if you include a option that forces Backup Server to rewind a tape before use, it always rewinds the tape to the beginning instead of reading the tape from the point of the dump.

Listing the current device names

To list the backup devices for your system, run:

```
select * from master..sysdevices
```

```
  where status = 16 or status = 24
```

To list both the physical and logical names for database and backup devices, use sp_helpdevice:

```
sp_helpdevice tapedump1
device_name physical_name
 description
 status cntrltype vdevno vpn_low      vpn_high
 ------ --------- ------------ -------- -------
tapedump1 /dev/nrmt4
tape, 625 MB, dump device
   16          3            0        0   20000
```

Adding a backup device

Use sp_addumpdevice to add a backup device:

sp_addumpdevice{ "tape" | "disk"} , *logicalname*, *physicalname*, *tapesize*

where:

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 the Adaptive Server current working directory.

tapesize is the capacity of the tape in megabytes. Most 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.

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 will 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 must 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 you upgrade a user database to the current version of Adaptive Server, 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, create index 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, load transaction may take as long to reconstruct the index as create index 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:

- Unlogged writetext
- select into on a permanent table
- Fast bulk copy (bcp) into a table with no triggers or indexes

You cannot 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 before executing any of these commands.

Dumping a database when the log has been truncated

dump transaction with truncate_only and dump transaction with no_log remove transactions from the log without making a backup copy. To ensure recoverability, dump the database each time you run either command because of lack of disk space. You cannot copy the transaction log until you have done so. See "Using the special dump transaction options" on page 312 for more information.

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. If this happens, you must dump the entire database—not the transaction log—to ensure recoverability.

Scheduling backups of *master*

Back up the master database *regularly and frequently.*

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.

Dumping *master* after each change

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. 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 procedures 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.

You cannot use the dataserver command to automatically recover changes that result from these commands. 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 27.

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 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 reenter all the changes you have made 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. Always use dump database to back up the model database and dump transaction with truncate_only to purge the transaction log after each database dump.

Scheduling backups of the *sybsystemprocs* database

The sybsystemprocs database stores only system procedures. 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 installmaster, then reenter 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 13, "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 sybsystemprocs. 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, truncate the log when you dump the database.

If you are running on a UNIX system or 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 sp_volchanged, and you cannot use this procedure on a server while sybsystemprocs is in the process of recovery.

Configuring Adaptive Server for simultaneous loads

Adaptive Server can perform multiple load and dump commands simultaneously. Loading a database requires one 16K buffer for each active database load. By default, Adaptive Server is configured for six simultaneous loads. 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 parameter requires a restart of Adaptive Server. See see Chapter 5, "Setting Configuration Parameters," for more information. These buffers are not used for dump commands or for load transaction.

Gathering backup statistics

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 load database and apply successive transaction log dumps with load transaction.

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 are.

After 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 are much better prepared to get your databases online if disaster strikes.

CHAPTER 12

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.

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:

- compress= to compress your dump files to a local file.
- 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

Note When a user database is dumped, its database options are not dumped because they are stored in the sysdatabases table of the master databases. This is not a problem if you load a previously dumped database onto itself because rows in sysdatabases describing this database still exist in master. However, if you drop the database before you perform the load database, or if you load the database dump on a new server, these database options are not restored. To restore the image of a user database, you must also re-create the database options.

Table 12-1 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 12-1: Syntax for routine dumps and log dumps after device failure

Information provided	Routine database or log dump	Log dump after data segment device failure
Command	`dump {database | transaction}`	`dump tran[saction]`
Database name	*database_name*	*database_name*

Information provided	Routine database or log dump	Log dump after data segment device failure
Compression to a local file (supported for backward compatibility)	to [compress::[*compression_level*::]]	to [compress::[*compression_level*::]]
Dump device	*stripe_device*	*stripe_device*
Remote Backup Server	[at *backup_server_name*]	[at *backup_server_name*]
Tape device characteristics	[density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,	[density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,
Volume name for single device	dumpvolume = *volume_name*	dumpvolume = *volume_name*
File name for single device	file = *file_name*]	file = *file_name*]
Characteristics of additional devices (up to 31 devices; one set per device)	[stripe on [compress::[*compression_level*::]] stripe_device [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name* file = *file_name*]]...	[stripe on [compress::[*compression_level*::]] stripe_device [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, 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*, capacity = *number_kilobytes*,
Compression to a remote server (recommended syntax for compressed dumps)	compression = *compress_level*	compression = *compress_level*
Volume name for all devices	dumpvolume = *volume_name*	dumpvolume = *volume_name*
File name for all devices	file = *file_name*,	file = *file_name*,
Tape handling options	[nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init],	[nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init],

Information provided	Routine database or log dump	Log dump after data segment device failure
Password; not available for dump tran or load tran	`passwd = password,`	
Option to dump only completed transactions; not available for dump database	`standby_access`	
Option to not truncate log		`no_truncate`
Message destination	`, notify = {client | operator_console}}]`	`, notify = {client | operator_console}}]`

Table 12-2 shows the syntax for loading a database, applying transactions from the log, and returning information about dump headers and files.

Table 12-2: Syntax for load commands

Information provided	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*
Compression to a local file. Supported for compatibility with older scripts and dumps.[1]	`from [compress::]`	`from [compress::]`
Dump device	*stripe_device*	*stripe_device*
Remote Backup Server	`[at backup_server_name]`	`[at backup_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 [compress::]stripe_device [at backup_server_name] [density = density, dumpvolume = volume_name, file = file_name]...`	`[stripe on [compress::]stripe_device [at backup_server_name] [density = density, dumpvolume = volume_name, file = file_name]...`

Information provided	Load database or apply recent transactions	Return header or file information but do not load backup
Characteristics of all dump devices	[with{ density = *density*, blocksize = *number_bytes*,	[with{ density = *density*, blocksize = *number_bytes*,
Compression to a remote server.	compression,	compression,
Volume name for all devices	dumpvolume = *volume_name*,	dumpvolume = *volume_name*,
File name for all devices	file = *file_name*,	file = *file_name*,
Tape handling options	[nodismount | dismount], [nounload | unload],	[nodismount | dismount], [nounload | unload],
Password name, not available for load tran	passwd = *password*,	passwd = *password*,
Lists dump files		listonly [= full],
Provides header information		headeronly,
Message destination	notify = {client | operator_console}]]	notify = {client | operator_console}
Loads log up to a specified time in the log; load tran only	until_time = datetime}]]	until_time = datetime}]]

[1]The older compress:: option is supported for backward compatibility. Sybase recommends that you use the compress= option during dump tran or dump database. No corresponding option is required during a load.

Table 12-3 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 12-3: Special dump transaction options

Information provided	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, see "Designating responsibility for backups" on page 334.

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 12-4 shows the syntax for backing up and loading a database or log.

Table 12-4: Indicating the database name and dump device

	Backing up a database or log	**Loading a database or log**
Database name	`dump {database | tran} database_name`	`load {database | tran} database_name`
Compress (supported for backward compatibility)	`to [compress::[compression_level:::]]`	`from [compress::]`
Dump device	`stripe_device`	`stripe_device`

	Backing up a database or log	Loading a database or log
	[at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_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*, compression = *compress_level*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init], passwd = *password*, standby_access [notify = {client | operator_console}]}]	[at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, [notify = {client | operator_console}] until_time = datetime}]]

Rules for specifying database names

You can specify the database name 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! Do not 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

When you specify 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*; 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 the Adaptive Server 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.
- If you dump a transaction log using with standby_access, you must load the dump using with standby_access.

Examples

The following examples use a single tape device for dumps and loads. You need not use the same device for dumps and loads.

- On UNIX:

```
dump database pubs2 to "/dev/nrmt4"
load database pubs2 from "/dev/nrmt4"
```

- 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

When you issue a dump database or dump transaction command, Backup Server checks whether 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.
```

To configure the device, issue the dump database or dump transaction with the init parameter. 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 sevice 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 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

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.

Compressing a dump

The dump command includes two options that allow you to compress databases and transaction logs using Backup Server, thereby reducing your space requirements for your archived databases. The parameters are:

- compress::[*compression_level*::] – compresses to a local file. Causes the Backup Server to invoke a external filter, and is supported for backward compatibility.

- with compress=*compress_level* – compresses to a remote server. Causes the Backup Server to use its own native compression method. This is the preferred compression option.

Table 12-5 shows the syntax for compress= and with compress= commands:

Table 12-5: Indicating the database name and dump device

	Backing up a database or log	**Loading a database or log**
	dump {database | tran} *database_name* to	load {database | tran} *database_name* from
Compress to local file (supported for backward compatibility)	[compress::[*compression_level* ::]]	[compress::]
	stripe_device [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on	*stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on
	[compress::[*compression_level* ::]] *stripe_device* [at *backup_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*,	[compress::] *stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*,
Compress to remote machine	compression = compress_level,	compression,

Backing up a database or log	Loading a database or log
dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, retaindays = *number_days*, [noinit | init], standby_access [notify = {client | operator_console}]}]	dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, [notify = {client | operator_console}] until_time = datetime}]]

Syntax

The partial syntax specific to dump database … compress= and dump transaction … compress= is:

dump database *database_name*
to compress=[*compression_level*] *stripe_device*
…[stripe on compress::[*compression_level*::]*stripe_device*] …

dump transaction *database_name*
to compress=[*compression_level*] *stripe_device*
…[stripe on compress::[*compression_level*::]*stripe_device*]…

Where:

- *database_name* – is the database you are loading into
- compress=*compression_level* – is a number between 0 and 9, with 0 indicating no compression, and 9 providing the highest level of compression. If you do not specify *compression_level*, Adaptive Server does not compress the dump.
- compress::*compression_level* – is included to support backward compatibility. The compress=*compression_level* option is the preferred syntax to compress a dump.
- *stripe_device* – is the full path to the archive file of the database or transaction log you are compressing. If you do not include a full path for your dump file, Adaptive Server creates a dump file in the directory in which you started Adaptive Server.

Use the stripe on clause to use multiple dump devices for a single dump. See "Specifying additional dump devices: the stripe on clause" on page 380 for more information about the stripe on clause.

Note The older compress:: option works only with local archives; you cannot use the servername option. To compress to a remote machine, you must use the preferred compress=*compress_level* option.

The compress= parameter of the dump command allows you to reduce your space requirements for your archived databases. With Adaptive Server 12.5.2 and later, the compress= parameter enables you to compress your dumps to a remote machine.

If you use the older compress:: option, you need not include the compression level when you load the database dump. However, you can issue load with listonly=full to determine the compression level at which the dump was made.

If you use the native compress= option, you need not include the compress= option when you load the database dump.

The partial syntax for dump database is:

```
dump database database_name to file_name
    [ with compression=compress_level]
```

where:

- *database_name* – is the name of the database from which you are copying data. The database name can be specified as a literal, a local variable, or a stored procedure parameter.
- *file_name* – is the name of the dump file. The name cannot exceed 17 characters and must conform to operating system conventions for file names. The pathname cannot exceed 255 characters.
- *compress_level* – is a number between 1 and 9, with 9 providing the highest level of compression. There is no default compression level; if you do not specify a *compress_level*, Adaptive Server does not compress the dump.

Examples

Example 1 Dumps the pubs2 database into a compressed file:

```
dump database pubs2 to device_options...compress=4
```

The *compression_level* must be a number between 0 and 9. The compress= option does not recognize numbers outside this range. For example, the following syntax creates a file which is compressed with a compression level of 9:

```
dump database pubs2 to device_options...compress=9
```

In general, the higher the compression numbers, the smaller your archives are compressed. However, the compression result depends on the actual content of your files.

Table 12-6 shows the compression levels for the pubs2 database. These numbers are for reference only; the numbers for your site may differ depending on OS level and configuration.

Table 12-6: Compression levels and compressed file sizes for pub2

Compression levels	Compressed file size
No compression/Level 0	630K
Level 1	128K
Level 2	124K
Level 3	121K
Level 4	116K
Level 5	113K
Level 6	112K
Level 7	111K
Level 8	110K
Level 9	109K

The higher the compression level, the more CPU-intensive the process is.

For example, you may not want to use a level-9 compression when archiving your files. Instead, consider the trade-off between processing effort and archive size. Compression level 6 provides optimal CPU usage, producing an archive that is 60 – 80 percent smaller than a regular uncompressed archive. Sybase recommends that you initially use compression level 6, then increase or decrease the level based on your performance requirements.

Example 2 Dumps the pubs2 database to the remote machine called "remotemachine" and uses a compression level of 4:

```
dump database pubs2 to "/Syb_backup/mydb.db" at remote_machine
with compression =”4”
```

For complete syntax information about dump database and dump transaction, see the *Reference Manual: Commands*.

Backup Server dump files and compressed dumps

Note This section describes issues pertinent to the compress:: option. They do not apply to the preferred compress= option.

When you perform dump database or dump transaction to a tape device using an archive file that already exists, Backup Server automatically checks the header of the existing dump archive. If the header is unreadable, Backup Server assumes that the file is a valid non-archive file, and prompts you to change the dump archive:

```
Backup Server: 6.52.1.1: OPERATOR: Volume to be overwritten on
'/opt/SYBASE/DUMPS/model.dmp' has unrecognized label data.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
    @session_id = 5,
    @devname = '/opt/SYBASE/DUMPS/model.dmp',
    @action = { 'PROCEED' | 'RETRY' |
'ABORT' },
    @vname = <new_volume_name>
```

For this reason, if you perform dump database or dump transaction to a file without the compress:: option into an existing compressed dump archive, Backup Server does not recognize the header information of the archive because it is compressed.

Example

The second dump database reports an error, and prompts you with sp_volchanged:

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
go
```

To prevent this error, include the with init option in your subsequent dump database and dump transaction commands:

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
   with init
go
```

Loading compressed dumps

Note This section describes issues pertinent to the compress:: option. If you make a dump with the native compress= option, it does not require any specific syntax to load..

If you use dump ... compress:: to dump a database or transaction log, you must load this dump using the load ... compress:: option.

The partial syntax for load database ... compress:: and load transaction ... compress:: is:

```
load database database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…

load transaction database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…
```

The *database_name* in the syntax is the database you archived, and compress:: invokes the decompression of the archived database or transaction log. *archive_name* is the full path to the archived database or transaction log that you are loading. If you did not include a full path when you created your dump file, Adaptive Server created a dump file in the directory in which you started Adaptive Server.

If you use the compress:: option, it must be part of the stripe on clause for each dump device. If you use the compress= option, it is used once after the device list. See "Specifying additional dump devices: the stripe on clause" on page 380 for more information about the stripe on clause.

Note Do not use the *compression_level* variable for the load command.

For complete syntax information about load database and load transaction, see the *Reference Manual: Commands*.

Specifying a remote Backup Server

Use the at *backup_server_name* clause to send dump and load requests over the network to a Backup Server running on another machine.

Table 12-7 shows the syntax for dumping or loading from a remote Backup Server.

Table 12-7: 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 [compress::[*compression_level*::]] *stripe_device*	load {database | tran} *database_name* from [compress::] *stripe_device*
Remote Backup Server	[at *backup_server_name*]	[at *backup_server_name*]
	[density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_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*, compression = *compress_level*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, retaindays = *number_days*, [noinit | init], standby_access [notify = {client | operator_console}]}]	[density = *density*, blocksize = *number_bytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, compression, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, [notify = {client | operator_console} until_time = *datetime*}]]

Note The compress:: option works only with local archives; you cannot use the *backup_server_name* option.

Sending dump and load requests over the network 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 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
```

The *backup_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 *backup_server_name* must be the same in both the local and the remote interfaces file.

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; *Sybase recommends that you use them*.

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 that you specify using the with clause.

Table 12-8 shows the syntax for specifying the tape density, block size, and capacity.

Table 12-8: Specifying tape density, block size, and capacity

Backing up a database or log	Loading a database or log
dump {database | tran} *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*]	load {database | tran} *database_name* from [compress::]*stripe_device* [at *backup_server_name*]

	Backing up a database or log	Loading a database or log
Characteristics of a single tape device	[density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,	[density = *density*,
	dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] ...]	dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...]
Characteristics of all dump devices	[with{ density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,	[with{ density = *density*,
	dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init], [notify = {client | operator_console}] standby_access}]	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.

This option has no effect 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 "Reinitializing a volume before a dump" on page 385.

Overriding the default block size

The blocksize parameter specifies the number of bytes per I/O operation for a dump device. By default, the dump and load commands choose the "best" block size for your operating system. Whenever 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 UNIX systems, the block size specified on the load command is ignored. Backup Server uses the block size that was used to make the dump.

Specifying a higher block size value

If you dump to a tape using the dump database or dump transaction commands, and specify a block size value that is higher than the maximum blocksize of a device as determined by Backup Server, the dump or the load may fail on certain tape drives. An operating system error message displays; for example, on an 8mm tape drive on HP the error message is:

```
Backup Server: 4.141.2.22: [2] The 'write' call
failed for device 'xxx' with error number 22 (Invalid
argument). Refer to your operating system
documentation for further details.
```

Do not specify a block size greater than the device's block size stored in the tape device configuration file in *$SYBASE/backup_tape.cfg*. The block size for a device is the fifth field of the line in the tape device configuration file.

This error occurs only on tape drives where tape auto config is run; that is, the device models are not hard-coded in Backup Server code.

Specifying tape capacity for dump commands

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 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 devices.

Non-rewinding tape functionality for Backup Server

The non-rewinding tape functionality automatically positions the tape at the end of valid dump data, which saves time when you want to perform multiple dump operations.

Dump label changes

Backup Server writes an End-of-File label, EOF3, at the end of every dump operation.

Note You cannot load a tape with this label into any version of Adaptive Server earlier then 12.0.

Tape operations

When a new dump is performed, Backup Server performs a scan for the last EOF3 label.

If the EOF3 label is found, the pertinent information is saved and the tape is positioned forward to the beginning of the next file on tape. This is the new append point.

If the EOF3 label is not found or any other problem is encountered, Backup Server rewinds the tape and scans forward. Any error that occurs during these steps does not abort the dump operation, but causes Backup Server to default to rewind-and-scan behavior. If the error persists during the rewind and scan, the dump command aborts.

Dump version compatibility

Backup Server activates non-rewinding logic only if the label version on the tape is greater than or equal to 5. Therefore, a dump command with the with init clause is needed to activate this logic. If a dump without init is initiated onto a volume with a label version less than 5, you are prompted to change the volume, and the dump starts on the next volume. The label version of a multi-volume dump does not change in the middle of the volume set.

Table 12-9 defines the label versions for which the new behavior is enabled.

Table 12-9: Label version compatibility

Label version	Enabled
'3'	No
'4'	No
'5'	Yes
'6'	Yes

Specifying the volume name

Use the with dumpvolume = *volume_name* option to specify the volume name. dump database and dump transaction write the volume name to the SQL tape label. load database and load transaction 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 12-10 shows the syntax for specifying a volume name.

Table 12-10: Specifying the volume name

	Backing up a database or log	**Loading a database or log**

	`dump {database | tran}`
	`  database_name`
	` to`
	`[compress::[compression_level::]]`
	`  stripe_device`
	`  [at server_name]`
	`  [density = density,`
	`  blocksize = number_bytes,`
	`  capacity = number_kilobytes,`
		`load {database | tran}`
		`  database_name`
		` from`
		`[compress::]stripe_device`
		`  [at server_name]`
		`  [density = density,`

Volume name for single device	`  dumpvolume = volume_name,`	`  dumpvolume = volume_name,`

	`  file = file_name]`
	` [stripe on`
	`  [compress::[compression_level::]]`
	`  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`
		`  [compress::]stripe_device`
		`  [at server_name]`
		`  [density = density,`
		`  dumpvolume = volume_name,`
		`  file = file_name]...]`
		` [with {`
		`  density = density,`

Volume name for all devices	`  dumpvolume = volume_name,`	`  dumpvolume = volume_name,`

	`  file = file_name,`
	`  [nodismount | dismount],`
	`  [nounload | unload],`
	`  retaindays = number_days,`
	`  [noinit | init],`
	`  [notify = {client`
	`  |operator_console}]`
	`  standby_access}]`
		`  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 load database.

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.

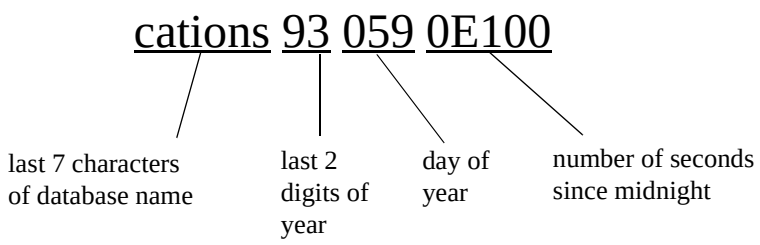
Table 12-11 shows the syntax for specifying the name of a dump.

Table 12-11: Specifying the file name for a dump

	Backing up a database or log	Loading a database or log
	dump {database | tran} *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*,	load {database | tran} *database_name* from [compress::] *stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*,
File name for single device	file = *file_name*]	file = *file_name*]
	[stripe on [compress::[*compression_level*::]] *stripe_device* [at *server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes,* dumpvolume = *volume_name*,	[stripe on [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*,
File name for additional devices	file = *file_name*]	file = *file_name*]
	[with{ density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*,	[with{ density = *density,* dumpvolume = *volume_name*,
File name for all devices	file = *file_name*]	file = *file_name*]
	[nodismount | dismount], [nounload | unload], retaindays = *number_days,* [noinit | init], passwd = *password*, [notify = {client | operator_console}] standby_access}]	[nodismount | dismount], [nounload | unload], passwd = *password*, [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, the date, and time the dump was made:

Figure 12-1: 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. Label each backup tape with the volume name and file name before storing it.

When you load a database or transaction log, 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"
```

Improving dump or load performance

When you start Backup Server, you can use the -m parameter to improve the performance of the dump and load commands by configuring more shared memory for the Backup Server. The -m parameter specifies the maximum amount of shared memory used by the Backup Server. You must also configure your operating system to ensure that this amount of shared memory is available to the Backup Server. After a dump or load operation is completed, its shared memory segments are released.

Note Configuring more shared memory improves dump and load performance only if the performance limits of the hardware setup have not been reached. Increasing the value of -m may not result in improved performance when dumping to a slow tape device such as QIC, but it can improve performance significantly when dumping to a faster device, such as DLT.

Compatibility with prior versions

There are some compatibility issues between dump files and Backup Server. Table 12-12 indicates the dump file formats that can be loaded by your currently running version and previous versions of local Backup Servers.

Table 12-12: Server for local operations

	New dump file format	Old dump file format
Current version of server	Yes	Yes
Earlier version of server	No	Yes

Table 12-13 and Table 12-14 indicate the dump file formats that can be loaded by the current and prior versions of remote Backup Servers. In a remote Backup Server scenario, the master server is the Backup Server on the same machine as the database and Adaptive Server, and the slave server is the Backup Server on the same remote machine as the archive device.

Table 12-13 indicates the load operations that work when master server is the current version of Backup Server.

Table 12-13: New version of master server

	New dump file format	Old dump file format
New slave version of server	Yes	Yes
Prior slave version of server	No	Yes

Table 12-14 indicates the load operations that work when the master server is a prior version.

Table 12-14: Prior version of master server

	New dump file format	Old dump file format
New slave version of server	No	Yes
Prior slave version of server	No	Yes

Labels stored in integer format

Backup Server 12.0 and later store the stripe number in integer format. Earlier versions of Backup Server stored the 4-byte stripe number in the HDR1 label in ASCII format. These earlier versions of Backup Server cannot load a dump file that uses the newer dump format. However, Backup Server version 12.0 and later can read and write earlier versions of the dump format.

When performing a dump or load operation involving one or more remote servers, the operation aborts with an error message, if:

- The versions of one or more of the remote Backup Servers are earlier than 12.0, and the database is dumped to or loaded from more than 32 stripes, or:
- The dump file from which one or more of the Backup Servers are reading during a load is from an earlier version's format, and the number of stripes from which the database is loaded is greater than 32.

Configuring system resources

Before you perform dumps and loads. you must configure the local and remote Backup Servers at start-up by providing the appropriate values for the system resources controlled by the command line options. See the *Utility Guide* for a complete list of the command line options.

If your system resources are not configured properly, the dump or load may fail. For example, a remote dump to greater than 25 stripes with the local and remote Backup Servers started with default configuration fails because the maximum number of network connections that Backup Server can originate (specified by the -N option) is 25; however, by default the maximum number of server connections into the remote Backup Server (specified by the -C option) is 30.

To configure the system to use the higher stripe limitations, set the following operating system parameters:

- Number of shared memory segments to which a process can attach
- Number of shared memory identifiers
- Swap space

If these parameters are not configured properly, when a dump is started to (or a load is started from) a large number of stripes, the operation may abort because of lack of system resources. In this case, you receive a message that Backup Server could not create or attach to a shared memory segment and therefore the SYBMULTBUF processes are terminated.

Setting shared memory usage

The syntax for starting Backup Server with the -m parameter is, where *nnn* is the maximum amount of shared memory in megabytes that the Backup Server can use for all of its dump or load sessions:

```
backupserver [-m nnn]
```

The -m parameter sets the upper limit for shared memory usage. However, Backup Server may use less memory than specified if it detects that adding more memory will not improve performance.

Backup Server determines the amount of shared memory available for each stripe by dividing the -m value by the configured number of service threads (-P parameter).

The default value for -m is the number of service threads multiplied by 1MB. The default value for -P is 48, so the default maximum shared memory utilization is 48MB. However, Backup Server reaches this usage only if all the 48 service threads are active concurrently. The maximum value for -P is the maximum number of service threads, 12,288. For more information about -P, see “Configuring user-defined roles” on page 392.

The amount of shared memory per stripe available for Backup Server is inversely proportional to the number of service threads you allocate. If you increase the maximum number of service threads, you must increase the -m value also, to maintain the same amount of shared memory per stripe. If you increase the -P value but do not increase the -m value, the shared memory allocated per stripe can decrease to the point that the dump or load cannot be processed.

To determine how much to increase the -m value, use this formula:

(-m value in MB) * 1024/(-P value)

If the value obtained by this formula is less than 128KB, Backup Server cannot start.

The minimum value for -m is 6MB. The maximum value for -m depends on operating system limits on shared memory.

If you create a dump using a Backup Server with a high shared memory value, and attempt to load the dump using a Backup Server with a lower shared memory value, Backup Server uses only the available memory. This results in degradation of load performance.

If the amount of shared memory available per stripe at load time is less than twice the block size used at dump time, Backup Server aborts the load with an error message.

Setting maximum number of stripes

The maximum number of stripes that Backup Server can use is limited by the maximum number of Open Server threads it can create. Open Server imposes a maximum limit of 12K on the number of threads an application can create.

Backup Server creates one service thread for each stripe. Therefore, the maximum number of local stripes Backup Server can dump to or load from is 12,286.

As an additional limitation, Backup Server uses two file descriptors for each stripe, apart from the file descriptors associated with the error log file, interfaces file, and other system files. However, there is a per-thread limitation imposed by the operating system on the number of file descriptors. Open Server has a limitation of 1280 on the number of file descriptors that an application can keep track of.

The formula for determining the approximate maximum number of local stripes to which Backup Server can dump is:

$$\frac{\text{(The smaller of either the OS limitation or the OpenServer limitation)} - 2}{2}$$

The formula for determining the approximate maximum number of remote stripes to which Backup Server can dump is:

$$\frac{\text{(The smaller of either the OS limitation or the OpenServer limitation)} - 2}{3}$$

For details about the default and maximum file descriptor limits, see your operating system documentation.

Setting maximum number of network connections

The maximum number of network connections a local Backup Server can originate is limited by Open Server to 9118. Because of this, the maximum number of remote stripes that Backup Server can use in a single dump or load operation is 9118.

A remote Backup Server accepts a maximum of 4096 server connections at any one time. Therefore, the maximum number of remote stripes to a single remote Backup Server is 4096.

Setting maximum number of service threads

The -P parameter for Backup Server configures the number of service threads Open Server creates. The maximum number of service threads is 12,228. The minimum value is 6. The maximum number of threads equals the maximum number of stripes available. If you have started Backup Server without setting a high enough -P value, and you attempt to dump or load a database to a number of stripes that exceeds the number of threads, the dump or load operation fails.

Specifying additional dump devices: the *stripe on* clause

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 multiple stripe on clauses.

Table 12-15 shows the syntax for using more than one dump device.

Table 12-15: Using more than one dump device

	Backing up a database or log	**Loading a database or log**

	`dump {database | tran}`
	`  database_name`
	` to`
	`[compress::[compression_level::]]`
	`  stripe_device`
	`  [at backup_server_name]`
	`  [density = density,`
	`  blocksize = number_bytes,`
	`  capacity = number_kilobytes,`
	`  dumpvolume = volume_name,`
	`  file = file_name]`
		`load {database | tran}`
		`  database_name`
		` from [compress::]stripe_device`
		`  [at backup_server_name]`
		`  [density = density,`
		`  dumpvolume = volume_name,`
		`  file = file_name]`

Characteristics of an additional tape device (one set per device; up to 31 devices)	`[stripe on`
	`  [compress::[compression_level::]`
	`]`
	`  stripe_device`
	`  [at server_name]`
	`  [density = density,`
	`  blocksize = number_bytes,`
	`  capacity = number_kilobytes,`
	`  dumpvolume = volume_name,`
	`  file = file_name] ...]`
		`[stripe on`
		`  [compress::]stripe_device`
		`  [at server_name]`
		`  [density = density,`
		`  dumpvolume = volume_name,`
		`  file = file_name] ...]`

	Backing up a database or log	Loading a database or log
	[with{ density = *density*, blocksize = *number_bytes*, compression = *compress_level* capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password* retaindays = *number_days*, [noinit | init], [notify = {client | operator_console}] standby_access}]	[with{ density = *density*, compression = *compress_level* dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password* [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 a new tape 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 multiple 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
```

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 12-16 shows the syntax for tape handling options.

Table 12-16: Tape handling options

	Backing up a database or log	Loading a database or log
	dump {database | tran} *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, compression = *compress_level*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*,	load {database | tran} *database_name* from [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name* file = *file_name*] [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, compression, dumpvolume = *volume_name*, file = *file_name*,
Tape handling options	[nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init],	nodismount | dismount], [nounload | unload],
	passwd = *password*, standby_access} [notify = {client | operator_console}]]	passwd = *password*, [notify = {client | operator_console}]

Specifying whether to dismount the tape

On platforms that support logical dismounts, 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.

Protecting dump files from being overwritten

tape retention in days 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 you can set with sp_configure, 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, 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. You can append new dumps 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

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

To dump multiple databases to the same tape volume:

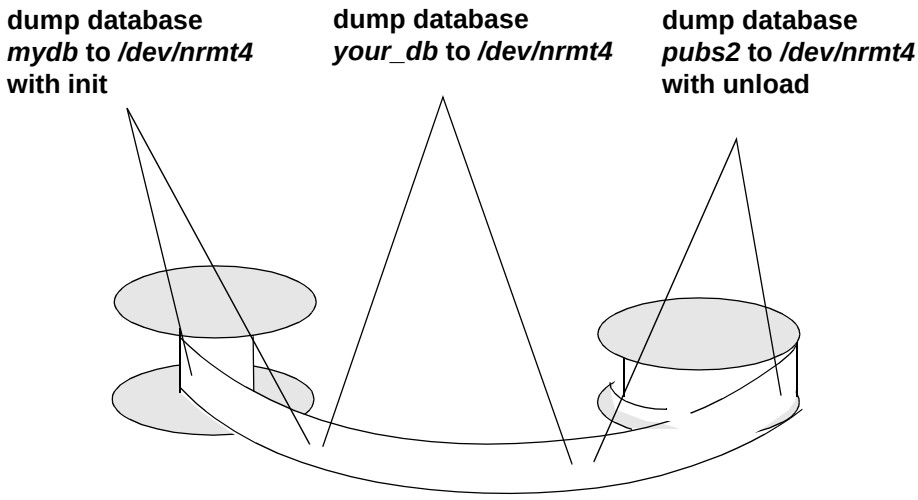
1 Use the init option for the first database. This overwrites any existing dumps and places the first dump at the beginning of the tape.

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 you dump the last database.

Figure 12-2 illustrates how use to dump three databases to a single tape volume.

Figure 12-2: Dumping several databases to the same volume



Dumping and loading databases with password protection

You can protect your database dump from unauthorized loads using the password parameter of the dump database command. If you include the password parameter when you make a database dump, you must also include this password when you load the database.

Table 12-17 shows the syntax for tape handling options.

Table 12-17: Password options

	Backing up a database or log	Loading a database or log
	dump {database | tran} *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, compression = *compress_level*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], retaindays = *number_days*,	load {database | tran} *database_name* from [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name* file = *file_name*] [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, compression, dumpvolume = *volume_name*, file = *file_name*, nodismount | dismount], [nounload | unload],
Password Option	passwd = *password*,	passwd = *password*,
	standby_access} [notify = {client | operator_console}]]	[notify = {client | operator_console}]

The partial syntax for the password-protected dump database and load database commands are:

dump database *database_name* to *file_name*
[with passwd = *password*]

load database *database_name* from *file_name*
[with passwd = *password*]

Where:

- *database_name* – is the name of the database that is being dump or loaded.
- *file_name* – is the name of the dump file.
- *password* – is the password you provide to protect the dump file from unauthorized users.

Your password must be between 6 and 30 characters long. If you provide a password that is less than 6 or greater than 30 characters, Adaptive server issues an error message. If you issue an incorrect password when you attempt to load the database, Adaptive Server issues an eror message and the command fails.

Examples

Example 1 Uses the password "bluesky" to protect the database dump of the pubs2 database:

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

Example 2 Loads the database dump using the same password:

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

Passwords and earlier versions of Adaptive Server

You can use the password-protected dump and load commands only with Adaptive Server version 12.5.2 and later. If you use the password parameter on a dump of a 12.5.2 version of Adaptive Server, the load fails if you try to load it on an earlier version of Adaptive Server.

Passwords and character sets

You can load the dump only to another server with the same character set. For example, if you attempt to load a dump from a server that uses an ASCII character set to a server that uses a non-ASCII character set, the load fails because the value of the ASCII password is different from the non-ASCII password.

Passwords entered by users are converted to Adaptive Server's local character set. Because ASCII characters generally have the same value representation across character sets, if a user's password is in an ASCII character set, the passwords for dump and load are recognized across all character sets.

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 that offer an operator terminal feature, volume change messages are always sent to an operator terminal on the machine where Backup Server is running. 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 12-18 shows the syntax for overriding the default message destination.

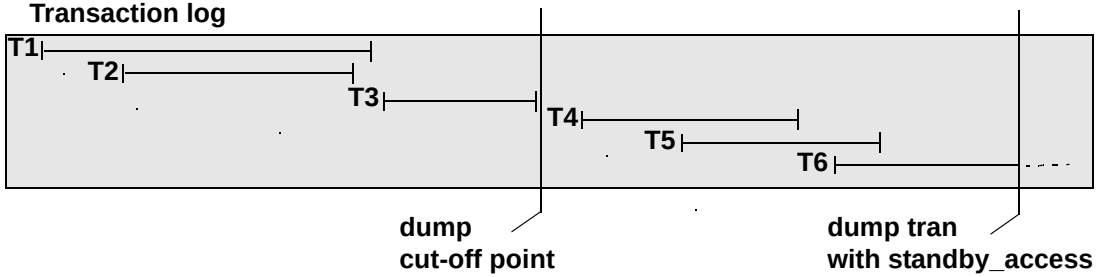
Table 12-18: Overriding the default message destination

	Backing up a database or log	Loading a database or log
	dump {database | tran} *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, compression = *compress_level*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init], passwd = *password*	load {database | tran} *database_name* from [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name* file = *file_name*] [stripe on [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, compression = *compress_level*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*,
Message destination	[notify = {client | operator_console}] standby_access}]	[notify = {client | operator_console}]

Bringing databases online *with standby_access*

with standby_access causes dump transaction to dump only completed transactions. It dumps the transaction log up to the point at which there are no active transactions. If you do not use with standby_access, the entire transaction log, including records for all open transactions is dumped. A transaction log dump using with standby_access is illustrated in Figure 12-3.

Figure 12-3: Dump cut-off point for dump transaction with standby_access



In Figure 12-3, a dump transaction...with standby_access command is issued at a point where transactions T1 through T5 have completed and transaction T6 is still open. The dump cannot include T5 because T6 is still open, and it cannot include T4, because T5 is still open. Thus, the dump must stop at the end of transaction T3, where it includes completed transactions T1 through T3.

Syntax

The syntax for with standby_access is:

```
dump tran[saction] database_name to...
    [with standby_access]
```

For more information about the with dump tran...with standby_access option, see the *Reference Manual*.

When do I use *with standby_access*?

Use dump tran[saction]...with standby_access when you are loading two or more transaction logs in sequence, and you want the database to be online between loads. For example, if you have a read-only database that gets its data by loading transaction dumps from a primary database. In this case, if the read-only database is used for generating a daily report based on transactions in the primary database, and the primary database's transaction log is dumped at the end of day, the daily cycle of operations is:

1 On the primary database: dump tran[saction]...with standby_access

2 On the read-only database: load tran[saction]...

3 On the read-only database: online database for standby_access

Warning! If a transaction log contains open transactions, and you dump it without using with standby_access, Adaptive Server does not allow you to load the log, bring the database online, and then load a subsequent transaction dump. If you are going to load a series of transaction dumps, you can bring the database online only after loading a dump originally made with standby_access, or after loading the entire series.

Bring databases online *with standby_access*

The online database command also includes a with standby_access option. Use for standby_access to bring a database online after loading it with a dump that was made using the with standby_access option.

Warning! If you try to use online database for standby_access with a transaction log that was not dumped using the with standby_access option, the command fails.

Syntax

The syntax for online database is:

online database *database_name* [for standby_access]

For more information about the with online database...for standby_access option, see the *Reference Manual*.

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 12-19 shows the syntax for finding the contents of a tape.

Table 12-19: Listing dump headers or file names

	Listing information about a dump
	load {database | tran} *database_name* from [compress::]*stripe_device* [at *backup_server_name*] [density = *density*, dumpvolume = *volume_name* file = *file_name*] [stripe on [compress::]*stripe_device* [at *server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, compression = *compress_level*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*,
List files on tape	listonly [= full],
List header information	headeronly [, file = *filename*],
	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                          '
```

The following is 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 use sp_volchanged to mount another volume and signal the volume change or to terminate the search operation for all stripe devices.

Copying the log after a device failure

Normally, dump transaction truncates the inactive portion of the log after copying it. Use with no_truncate to copy the log without truncating it.

no_truncate allows you to copy the transaction log after failure of the device that holds your data. It uses pointers in the sysdatabases and sysindexes tables to determine the physical location of the transaction log. You can use no_truncate 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 407.

Table 12-20 shows the syntax for copying a log after a device failure.

Table 12-20: Copying the log file after a device failure

	Copying with the no_truncate option
	dump transaction *database_name* to [compress::[*compression_level*::]] *stripe_device* [at *backup_server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] [stripe on [compress::[*compression_level*::]] *stripe_device* [at *backup_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*, compression = *compress_level*, dumpvolume = *volume_name*, file = *file_name*, [nodismount | dismount], [nounload | unload], passwd = *password*, retaindays = *number_days*, [noinit | init],
Do not truncate log	no_truncate,
	[notify = {client | operator_console}] standby_access}]

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 does not copy any data, with truncate_only 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 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.

with truncate_only 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, you must 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 does not copy any 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, you may fill the log completely, causing any further dump transaction commands to fail. Use alter database 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 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 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.

When 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 *Reference Manual*. For information about the last-chance threshold and threshold procedures, see Chapter 15, "Managing Free Space with Thresholds."

Responding to volume change requests

On UNIX and PC systems, use sp_volchanged to notify the Backup Server when the correct volumes have been mounted.

To use sp_volchanged, log in to 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 this 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 the file to load. If you do not specify a file name, 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*, 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. Each prompt includes 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 cannot use a database. Adaptive Server marks the database as suspect and displays a warning message. If the disk storing the master database fails, users cannot log in to the server, and users already logged in cannot perform any actions that access the system tables in master.

When your database device fails, Sybase recommends 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 After 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 *remaps* 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.

To examine and record the device allocations for all damaged databases:

1 In master, 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 blocks of data. Note the order, use, and size of each disk piece.

   For example, this is the output from a server that uses 2K logical pages translates into the sizes and uses described in Table 12-21:

```
segmap      size
-------     --------
     3        10240
     3         5120
     4         5120
     8         1024
     4         2048
```

Table 12-21: 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 7s, 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      status
-------    ------- ------  ----    -----------  -------------
mydb       46.0 MB sa        15    May 26 2005  no_options set

device_fragments  size    usage      created              free kbytes
----------------  -----   --------   ------------         -----------
datadev1          20 MB  data only  June 7 2005 2:05PM         13850
datadev2          10 MB  data only  June 7 2005 2:05PM          8160
datadev3           2 MB  data only  June 7 2005 2:05PM          2040
logdev1           10 MB  log only   June 7 2005 2:05PM  not applicable
logdev2            4 MB  log only   June 7 2005 2:05PM  not applicable
```

Dropping the databases

After you have performed the preceding steps *for all databases on the failed device*, use drop database 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 drop database, use the dropdb option of the dbcc dbrepair command:

```
dbcc dbrepair (mydb, dropdb)
```

If you are using a replicated database, use dbcc dbrepair to load a dump from a previous release of Adaptive Server to a more current version. For example:

- Loading a dump from a production system of an earlier release of Adaptive Server into a test system of the current release Adaptive Server, or,

- In a warm standby application, initializing a standby database of the current release of Adaptive Server with a database dump from an active database of an earlier release of Adaptive Server

See the *Error Message and Troubleshooting Guide* and *Reference Manual: Commands* 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 *Reference Manual* for more information.

Initializing new devices

Use disk init to initialize the new database devices. See Chapter 7, "Initializing Database Devices," for more 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 create database 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*. 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 12-21 on page 409, enter:

```
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 alter database 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 = "2M"
log on logdev1= "4M"
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 remaps 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*.

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.

If you dumped the transaction log using with standby_access, you must also load the database using standby_access.

After 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 load transaction. This is useful if, for example, a user inadvertently drops an important table; you can use until_time to recover the changes made to the database containing the table up to a time just before the table was dropped.

To use until_time 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 must 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 online database to make it available for use. In this example, the command to bring the mydb database online is:

```
online database mydb
```

Replicated databases

Before you upgrade replicated databases to the current version of Adaptive Server, 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, online database is automatically issued.

Upgrading to the current version of Adaptive Server

Refer to 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 online database to bring the databases online. Databases that are offline because they are in a load sequence are not automatically brought online by Replication Server.

Warning! Do not issue online database until all transaction logs are loaded.

Loading database dumps from older versions

When you upgrade an Adaptive Server installation to a new version, all databases associated with that server are automatically upgraded.

As a result, database and transaction log dumps created with an earlier version of Adaptive Server must be upgraded before they can be used with the current version of Adaptive Server.

Adaptive Server provides an automatic upgrade mechanism—on a per-database basis—for upgrading a database or transaction log made with Backup Sever to the current Adaptive Server version, thus making the dump compatible for use. This mechanism is entirely internal to Adaptive Server, 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 an older version of the master database. That is, if you upgraded Adaptive Server to the current version, you cannot load a dump of the master database from which you upgraded.
- Installing new or modified stored procedures. Continue to use installmaster.
- Loading and upgrading dumps generated earlier than SQL Server version 11.9.

How to upgrade a dump to Adaptive Server

To upgrade a user database or transaction log dump to the current version of Adaptive Server:

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 load database or load transaction. 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 online database, after the dump has successfully loaded, to activate the upgrade process.

Note Do *not* issue online database until after all transaction dumps are loaded.

Prior to SQL Server version 11.0, a database was automatically available at the end of a successful load sequence. With the current version of Adaptive Server, the user is required to bring the database online after a successful load sequence, using online database.

For dumps loaded from versions 11.9, 12.0, and 12.5, online database 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 the current version of Adaptive Server, no upgrade process is activated. You must still issue online database to place the database online—load database marks it as offline.)

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 *Reference Manual*.

3 After successful execution of online database, use dump database. 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. If this bit is set, it shows that the database is offline .

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, you can use these commands 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:

- Configuration upgrade version ID – shows the current version of Adaptive Server; it is stored in the sysconfigures table. sp_configure displays the current version of Adaptive Server as “upgrade version.”

- Upgrade version indicator – 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 – differentiates version 10.x logs from version 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

If you dump a database and load it onto a server with different cache bindings, be aware of cache bindings for a database and the objects in the database. 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 you made the dump.

When you bring a database online after recovery or by using online database 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 error log:

```
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, you can:

- 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

Use sp_helpcache to display 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 database 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. To change the name or location of a database when you reload it, use alter database 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 may 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:

1 Put both databases in single-user mode.

2 Drop the inconsistent referential constraint.

3 Check the data consistency with a query such as:

```
select foreign_key_col from table1
where foreign_key not in
(select primary_key_col from otherdb..othertable)
```

4 Fix any data inconsistency problems.

5 Re-create the constraint.

CHAPTER 13

Restoring the System Databases

This chapter explains how to restore the master, model, and sybsystemprocs databases.

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, recovery may include:

- Using load database to load backups of these databases,
- Using dataserver, installmaster, and installmodel to restore the initial state of these databases, or
- A combination of the above tasks.

To make the recovery of system databases as efficient as possible:

- Do not store user databases or any databases other than master, tempdb, model, and sybsystemdb 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.

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. Your master database is damaged if:

- Adaptive Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- dbcc reports damage during a regular check of your databases.

Recovering the *master* database

This section describes how to recover the master database and to rebuild the master device. It assumes:

- The master database is corrupt, or the master device is damaged.
- You have up-to-date printouts of the system tables, listed in "System and Optional Databases" on page 23.
- The master device contains *only* the master database, tempdb, model, and sybsystemdb.
- 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.

The *Error Message and 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 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 cannot use the system stored procedures.

Summary of recovery procedure

You 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 1: Find copies of system tables" on page 426
Shut down Adaptive Server, and use dataserver to build a new master database and master device.	"Step 2: Build a new master device" on page 426
Restart Adaptive Server in master-recover mode.	"Step 3: Start Adaptive Server in master-recover mode" on page 428
Re-create the master database's allocations in sysusages exactly.	"Step 4: Re-create device allocations for master" on page 430
Update the Backup Server network name in the sysservers table.	"Step 5: Check your Backup Server sysservers information" on page 430
Verify that your Backup Server is running.	"Step 6: Verify that your Backup Server is running" on page 431
Use load database to load the most recent database dump of master. Adaptive Server stops automatically after successfully loading master.	"Step 7: Load a backup of master" on page 432
Update the number of devices configuration parameter in the configuration file.	"Step 8: Update the number of devices configuration parameter" on page 432.
Restart Adaptive Server in single-user mode.	"Step 9: Restart Adaptive Server in master-recover mode" on page 432

Step	See
Verify that the backup of master has the latest system tables information.	"Step 10: Check system tables to verify current backup of master" on page 433
Restart Adaptive Server.	"Step 11: Restart Adaptive Server" on page 433
Check syslogins if you have added new logins since the last backup of master.	"Step 12: Restore server user IDs" on page 434
Restore the model database.	"Step 13: Restore the model database" on page 434
Compare 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 14: Check Adaptive Server" on page 435
Dump the master database.	"Step 15: Back up master" on page 435

Step 1: Find copies of system tables

Find copies of the 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 27.

Step 2: Build a new master device

You only need to build a new master device if your old master device is damaged beyond repair. Otherwise, you can recreate the master and model databases on your existing master device.

There are two procedures for recreating the master database: replacement of the master device, and forcing Adaptive Server to recreate the configuration area. Use the first procedure when only the master database corrupted. Use the second procedure if the master device's configuration area is also corrupt. You can often detect corruption in the configuration area because the server will refuse to run at all, complaining that this area is corrupt.

The following examples use the UNIX dataserver command. On Windows, use the sqlsrvr command.

Replacing the master device

Rebuild the master device with the dataserver -w option:

```
dataserver -w master
```

Note Your dataserver command may include other options such as command line flags specifying the device path name, server name, interfaces file name, and so on. These option are listed in the *RUN_servername* file that normally starts this server.

The -w master option causes Adaptive Server to search the device for database fragments belonging to the master database. After it finds these fragments, it writes a default master database into that space, then shuts down.

Restart Adaptive Server with the *RUN_servername* file.

Rebuilding the configuration area

If the configuration area is corrupt, you must use the -f option to force Adaptive Server to reconstruct this area. The syntax is:

```
dataserver -w master -f [-zpage_size] [-bdevice_size]
```

Note Your dataserver command may include other options such as command line flags specifying the device path name, server name, interfaces file name, and so on. These option are listed in the *RUN_servername* file that normally starts this server.

The following restrictions apply:

- You can specify the page size if it is wrong (for example, `-z8k`).
- You can specify the device size if it is wrong (for example, `-b125M`).
- Any allocation units on the disk that appear corrupt, or that are not currently allocated, are allocated to the master database.

Warning! Do *not* attempt to use this procedure to change your server's logical page size. Doing so does not work, and further corrupts the device to the point where you cannot recover it. Any device size you specify must be accurate; dataserver -w does not alter the device's size, but it may fail to find parts of some databases if the specified size is too small, and can fail entirely if the specified size is too large.

The -w master option, with or without -f, only recreates the master database, and All other allocation units on the disk are not changed, so you will probably be able to recover the data using 'disk refit' as described further down this chapter.

If the entire master device is corrupt (for example, in the case of a disk crash) you must replace the entire device by starting Adaptive Server in buildmaster mode:

1 If the existing master device is not on a raw partition and you plan to reuse it, remove the old master device file.

2 Run the server in buildmaster mode:

```
dataserver -zpage_size -bdevice_size [other options as appropriate]
```

Warning! You cannot use this command to change the server's page size at this time. All the other devices for this server use the configured page size, and will not recover properly if you use a different page size. However, since you are creating a new file, you can change the device size at this point.

When determining how large to make your master device, keep in mind that this device reserves 8 KB for its configuration area. For example, if you specify the device size as 96 megabytes, the final bit of space is wasted because there isn't enough space for a full allocation unit. You should add an additional .01 megabyte to the space you require for the device to account for this overhead. So, to have a full 96 megabyte, specify -b96.01M.

When you use this method to rebuild the master device, Adaptive Server creates new, default databases for master, model, tempdb, and sybsystemdb. These databases are all as small as possible for your installation's logical page size. If you intend to load backups to these databases, they may now be too small. You must increase their size with alter database before loading these backups.

Regardless of how your restore your master device, all users and passwords in your master database will be gone . The only remaining privileged login will be sa, with no password.

For details on dataserver, see the *Utility Guide*.

Step 3: Start Adaptive Server in master-recover mode

Start Adaptive Server in master-recover mode with the -m (UNIX and Windows NT) option.

- 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 Windows NT – start Adaptive Server from the command line using the sqlsrver command. Specify 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 Guide* 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 dataserver command on the master database, only the "sa" account exists, and its password is NULL.

Warning! Some sites have automatic jobs that log in to the server at start-up with the "sa" login. Be sure these are disabled.

Master-recover mode is necessary because the generic master database created with dataserver 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 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 4: Re-create device allocations for *master*

If you recreated your master device according to the process described in step 2 above, your master database may now be too small. Perform these steps to allocate more space for your master database:

1 From the hard copy version of sysusages, total the size values shown for dbid 1 (the dbid of the master database). Compare those to the size of the current master database. You can determine them by issueing:

```
select sum(size)
from sysusages
where dbid = 1
```

2 If your current master database is too small, use alter database to enlarge it so that it to the size you require. To convert logical pages to megabytes, use this query:

```
select N / (power(2,20) / @@maxpagesize)
```

Where *N* is the number of logical pages.

You should not need to alter the size of the master database if you rewrote the master database with the -m master option. Adaptive Server has recorded the allocation units used by all databases on the device, so you should already have as much space as you need to load your dump of master.

Note If you do not have a hard copy of sysusages, you can determine how much larger a database needs to be by attempting to load it. If it is too small, Adaptive Server displays an error message telling you how much larger to make it.

Previous versions of Adaptive Server required that you perform a complex series of steps to recreate allocations on the new master device the old version. This procedure is no longer necessary. Adaptive Server versions 15.0 and later perform most of the work for you.

Step 5: Check your Backup Server *sysservers* information

Log in to the server as “sa,” using a null password.

If the network name of your Backup Server is not SYB_BACKUP, 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 6: Verify that your Backup Server is running" on page 431.

If the reported srvnetname is *not* the same as the Backup Server in the interfaces file, update sysservers. The example below changes the Backup Server 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 verify that the correct row was modified and that it contains the correct value. If update 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 row, issue a commit transaction command.

Step 6: Verify that your Backup Server is running

On UNIX 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 Guide*.

On Windows NT, a locally installed Sybase Central and the Services Manager show whether Backup Server is running.

See the *Utility Guide* for the commands to start Backup Server.

Step 7: Load a backup of *master*

Load the most recent backup of the master database. Here are examples of the load commands:

- On UNIX platforms:

```
load database master from "/dev/nrmt4"
```

- On Windows NT:

```
load database master from "\\.\TAPE0"
```

See Chapter 12, "Backing Up and Restoring User Databases," for information on command syntax.

After load database completes successfully, Adaptive Server shuts down. Watch for any error messages during the load and shut down processes.

Step 8: 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 9: Restart Adaptive Server in master-recover mode" on page 432.

Configuration values are not available to Adaptive Server until after recovery of the master database, so instruct Adaptive Server to read the appropriate value for the number of devices parameter from a configuration file at start-up.

If your most recent configuration file is not available, edit a configuration file to reflect the correct value for the number of devices parameter.

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 reads the parameter values from the specified configuration file.

Step 9: Restart Adaptive Server in master-recover mode

Use startserver to restart Adaptive Server in master-recover mode (see "Step 3: Start Adaptive Server in master-recover mode" on page 428). 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 is now locked. Perform the rest of the recovery steps using an account with the System Administrator role.

Step 10: Check system tables to verify current backup of *master*

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 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 440.
- 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 440).

Note You must start Adaptive Server with trace flag 3608 before you run disk refit. However, make sure you read the information provided in the *Troubleshooting and Error Messages Guide* before you start Adaptive Server with any trace flag.

Step 11: Restart Adaptive Server

Restart Adaptive Server in normal (multiuser) mode.

Step 12: 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 sp_modifylogin 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 re redoing the logins, add placeholders for the skipped values and then drop them with sp_droplogin or lock them with sp_locklogin.

Step 13: 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/ASE-12_5/scripts
isql -Usa -Ppassword -Sserver_name < installmodel
```

  On Windows NT:

```
cd $SYBASE/ASE-12_5/scripts
isql -Usa -Ppassword -Sserver_name < instmodl
```

- Redo any changes you made to model.

Step 14: 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 15: 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 how to recover the model database when it is the only database that needs 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

dataserver can restore the model database without affecting master.

Warning! Shut down Adaptive Server before you use any dataserver command.

- On UNIX platforms:

```
dataserver -d /devname  -w model
```

- On Windows NT:

```
sqlsrvr -d physicalname -w model
```

Restoring *model* from a backup

If you can issue use model successfully, you can restore your model database from a backup with load database.

If you cannot use the database:

1 Follow the instructions for "Restoring the generic model database" on page 436.

2 If you have changed the size of model, reissue alter database.

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 436.

- 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 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*

This section assumes that your sybsystemprocs database exists, but is corrupt. If sybsystemprocs does not exist, skip this section; you must create it using create database.

To restore sybsystemprocs with *installmaster*:

1 Check to see which devices currently store sybsystemprocs. sp_helpdb probably will not work at this time, but the following query will:

```
select  lstart,
size / (power(2,20)/@@maxpagesize) as 'MB',
d.name as 'device name',
case when segmap = 4 then 'log'
when segmap & 4 = 0 then 'data'
else 'log and data'
end as 'usage'
from sysusages u, sysdevices d
where d.vdevno = u.vdevno
and d.status & 2 = 2
and dbid = db_id('sybsystemprocs')
order by 1
```

The result probably shows sybsystemprocs all on one disk fragment, and having `log and data` as its usage, but you may have a more complex layout than that. Save this query's results for later use.

2 Drop the database.

```
drop database sybsystemprocs
```

If that succeeds and the device is undamaged, go to step 3.

If the drop database fails, perform either:

- If sybsystemprocs is badly corrupted, the drop database may fail. In that case, you can remove the database by hand by deleting the information that identifies it:

```
sp_configure 'allow updates', 1
go
delete from sysusages
where dbid = db_id('sybsystemprocs')
delete from sysdatabases
where name = 'sybsystemprocs'
go
sp_configure 'allow updates', 0
go
```

- If the physical disk is damaged, you must drop the device:

```
sp_dropdevice name_of_sybsystemprocs_device
```

If you removed sybsystemprocs by hand or recreated the sybsystemprocs device, shut down Adaptive Server using shutdown with nowait. If you dropped the sybsystemprocs device and it was not a raw partition, remove the physical file. Restart Adaptive Server.

3 Recreate the sybsystemprocs device. If you dropped the sybsystemprocs device, create a new one via disk init. Then recreate sybsystemprocs using one of the methods below using the results you wrote down in step 1.

Note If you plan to load a backup copy of sybsystemprocs, you can include the for load option with the create database or alter database commands. However, you must use load database to load it before it can be used for any other purpose.

- If the displayed `usage` was all `log and data`, create a simple database using:

```
create database sybsystemprocs on device_name
= N
```

where *N* is the total size from all its previous sections. You may find that you need to create it on multiple devices to get the size you need.

- If the displayed usage contains any `log` or `data` entries, recreate this same layout using create database and alter database. You can group contiguous data or log sections on a single device, but avoid mixing log with data. Recreate the first group of data and log sections using create database:

  ```
  create database sybsystemprocs
  on device_1 = M
  log on device_2 = N
  ```

  where *M* is the sum of the first group of data sizes and *N* is the sum of the first group of log sizes. For each successive group, repeat this process using alter database instead of create database to enlarge the database.

4 Run the *installmaster* script to create the Sybase-supplied system procedures.

  On Unix platforms:

  ```
  isql -Usa -Ppassword -Sserver_name -i
  $SYBASE_/$SYBASE_ASE/scripts/installmaster
  ```

  On Windows (from the *%SYBASE%\%SYBASE_ASE%\scripts* directory):

  ```
  isql -Usa -Ppassword -S<server_name> -i instmstr
  ```

5 If your site added any procedures or made other changes in sybsystemprocs, you must make these changes.

Restoring *sybsystemprocs* with *load database*

If you write system procedures and store them in sybsystemprocs, there are 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 437. 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 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 Windows NT:

```
load database sybsystemprocs from "\\.\TAPE0"
```

Restoring system tables with *disk reinit* and *disk refit*

When 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 the procedures in this section to restore the proper information in the sysusages, sysdatabases, and sysdevices tables.

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 disk reinit. If you saved scripts from your original disk init commands, use them to determine the parameters for disk reinit (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 did 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 still need to know the value of vstart if you used a custom vstart in the original disk init command.

Table 13-1 describes the disk reinit parameters and their corresponding sysdevices data:

Table 13-1: 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*	Should be full path to device. Any relative paths are relative to the server's current working directory.
vdevno	vdevno	Select 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 error log 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, 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 dataserver.

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 7, “Initializing Database Devices,” or the *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. Check Adaptive Server with dbcc after running disk refit.

CHAPTER 14

Automatic Database Expansion

Automatic expansion of databases and devices lets you configure databases to expand automatically when they run out of space.

The automatic database expansion stored procedure sp_dbextend allows you to install thresholds that identify those devices with available space, and then appropriately alter the database—and the segment where the threshold was fired—on these devices.

Introduction

After you set up the database for automatic expansion, when a database grows to its free space threshold, internal mechanisms fire, increasing the size of the database by the amount of space your expansion policies specify. The automatic expansion process measures the amount of room left on all devices bound to the database. If there is sufficient room on the devices, the database continues to grow. If any devices are configured for expansion, those devices expand next. Finally, the database is expanded on those devices.

This automatic expansion process runs as a background task and generates informational messages in the server's error log about its progress.

Understanding disks, devices, databases, and segments

Figure 14-1 on page 446 shows the various layouts of physical resources that may exist in an Adaptive Server installation after a series of disk init, create database, and alter database operations. You can use this information to devise different plans of physical and logical space layout while testing stored procedures.

Raw disks can be partially occupied by a Sybase device. syb_device2 shows an entire raw disk fully occupied by a single Sybase device, on which multiple databases were created. On this raw disk (*/dev/vx/rdsk/sybase2/vol02*), there is still some empty space at the head of the device, which can happen when a database that initially occupied this space is subsequently dropped.

syb_device4 and syb_device5 show the layout of Sybase devices */agni4/syb_device4.dat* and */agni5/syb_device5.dat* on a file system disk, where the Sybase device occupies a portion of the disk, but there is still room for the device (an operating system file, for instance) to grow.

syb_device6 shows a Sybase file system disk that has fully occupied the entire available space on the physical disk, but still has unused space on the device. This space can be used to expand existing databases on this device.

These various devices illustrate database fragments for different databases. Each device for a particular database has one or more segments spanning that device.

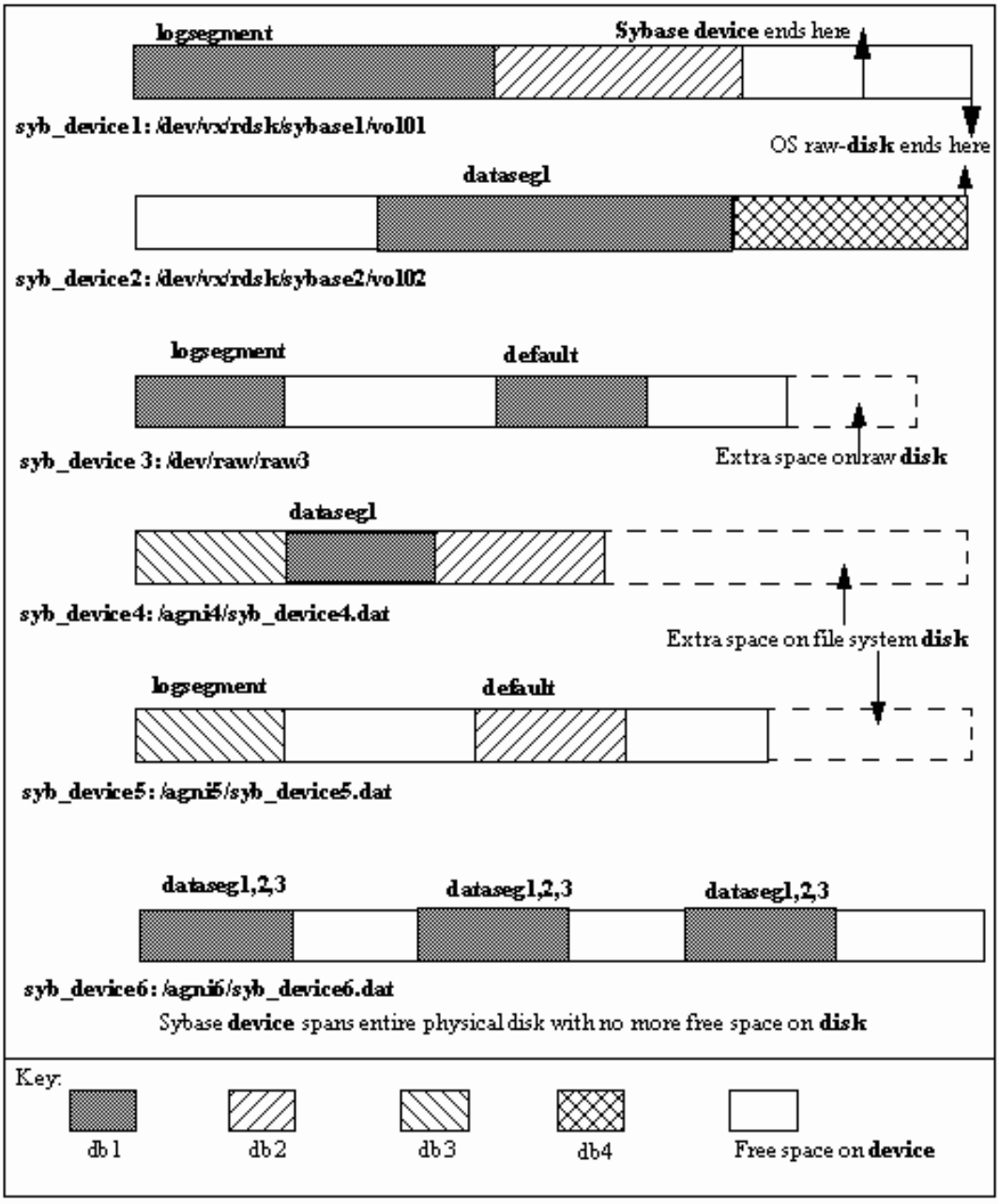
In syb_device6, */agni6/syb_device6.dat*, db1 spans three separate pieces of the device. That device also belongs to three different segments, data segments 1, 2, and 3. All three entries in sysusages for this database, db1, appear with a segment map value that includes all three segments.

However, on the device syb_device3 on */dev/raw/raw3*, the database has two pieces of the device, and one of them is exclusively marked as for the log segment while the other is marked as for the default segment. Assuming that an initial create database command has already been executed, the following SQL commands can produce this result:

```
alter database db1 on syb_device3 = "30M"
alter database db1 log on syb_device3 = "10M" with
      override
```

The first alter database command creates the database piece of default segment, and the second one creates the database piece of logsegment, forcing an override even though both pieces are on the same device. Space is expanded on a database in individual segments.

Figure 14-1: Database and device layouts



Threshold action procedures

Database expansion is performed by a set of threshold action procedures fired when free space crosses a threshold set for a segment. sp_dbextend is the interface for administering and managing the expansion process for a specified segment or device.

You can configure automatic expansion to run with server-wide default expansion policies, or you can customize it for individual segments in specified databases.You can install thresholds on key segments on which tables with critical data reside, allowing you a fine degree of control over how Adaptive Server meets the data space requirements for different kinds of tables. If your site has key tables with large volumes of inserts, these tables can be bound to specific segments, with site-specific rules for extending that segment. This enables you to avoid outages that can occur in a production environment with large loads on such key tables.

You cannot use the thresholds to shrink a database or its segment.

For more information on sp_dbextend, see the *Adaptive Server Reference Manual*.

Installing automatic database expansion procedures

Install the automatic expansion process using the *installdbextend* script, which loads rows into master.dbo.sysattributes describing defaults for automatic expansion in a database or in a device. The installation script also creates a control table in the model and tempdb databases.

If you are upgrading to Adaptive Server version 12.5.1 or later, you must install this script separately as part of your upgrade process.

❖ **Installing automatic database expansion**

1 Log in with sa_role permissions. In UNIX, *installdbextend* is located in the directory *$SYBASE/$SYBASE_ASE/scripts*. If you are running Windows NT, the location is *%SYBASE%/%SYABASE_ASE%/scripts*

2 Run *installdbextend* by entering:

```
isql -Usa -P -Sserver_name <$SYBASE/$SYBASE_ASE/scripts/installdbextend
```

The *installdbextend* installation script installs the family of threshold action procedures and sp_dbextend in the sybsystemprocs database.

Using the stored procedure

sp_dbextend allows you to customize the database and device expansion process, based on site-specific rules. Database administrators can configure the size or percentage by which each database and segment pair should be expanded, and the size or percentage by which each device should be expanded.

You can also limit the expansion of a database segment or device by specifying a maximum size beyond which no further expansion is possible.

You can use sp_dbexpand in a test mode, to simulate the expansion processes based on your chosen policies.

sp_dbextend provides ways to list the current settings and drop site-specific policy rules.

This information is stored as new attribute definitions in master.db.sysattributes.

Command options in the sp_dbextend interface

sp_dbextend has the following syntax:

```
sp_dbextend [ command [, arguments...] ]
```

where command is one of the options discussed below, and arguments specifies the database name, segment name, and amount of free space, among other things. For more information about sp_dbextend, see the *Reference Manual*.

The following command parameters define and specify user choices in sp_dbextend:

- set – sets the threshold at which a database and segment pair should fire. This command also allows you to specify the size and define the growth rate by which to expand the database and segment or device at each attempt, either in size values or as a percentage of the size of the device when the expansion is attempted.

  You can also use set to specify the maximum allowable size for the database, set the site-specific policy rules for expanding a device, and set the maximum size for the device.

- clear – clears any previously set rules of expansion for a specified database and segment, or for a specified device.

Using clear deletes the affected rows from master.db.systattributes, but automatic expansion is still in effect. Default rules for searching the devices a segment maps onto still apply when you consider where to expand the database, which device to expand, and in what order.

For instance, if you issue clear for one device, only that device continues as a candidate for expansion under the default expansion rules. Other devices expand according to their individual device characteristics.

To turn the automatic expansion feature completely off in a given database and the devices that database resides on, use the command 'clear', 'threshold'. This command executes sp_dropthreshold.

- modify – modifies site-specific policies, such as *growby* and *maxsize* in an existing entry for a database and segment. You can also use modify to modify system-supplied defaults.

  There is no support for modifying existing thresholds at which the database expansion threshold procedure fires. Use the existing interface sp_modifythreshold to make modifications.

- list – lists any existing rules for a specified database, segment, or device, and presents the data from master.db.sysattributes in a readable format.

  list allows you to view site-specific current policy rules in effect for each database, segment, or device previously configured by a set command. Databases, segments, and devices that do not appear in the output are subject to the default rules of expansion, which list also displays.

- listfull – lists fully the site-specific rules, including a comment column in the sysattributes table that displays a datetime stamp for when the rule was set and when it was last modified.

- check – examines current policies and verifies that they are consistent with the current space layout in each segment. If any policy settings appear redundant, ineffective, or incorrect, a warning message appears.

- simulate – simulates the database or device expansion schemes executed at runtime, according to the set of current policies implemented by the set command.

  You can perform these cycles of execution as many times as you like, to study the way database and disk expansion operate and to determine what pieces expand, and by how much space. To perform the expansion, use the execute parameter.

- execute – performs the database and segment, or device, expansion using the current set of policies.This option performs the expansion process immediately, irrespective of the current free space in the specified segment.
- reload [defaults] – reinitializes sysattributes with the system-supplied defaults for the *growby* and *maxsize* parameters, in all databases, segments, and devices, to revert to the original default behavior.

  For instance, you might make changes to system default rules using modify, and then run a simulation with simulate. Running reload [defaults] deletes any existing rows describing system default behavior and loads new rows. It does not change existing rows defining user-specified policies.
- help – provides help in using the procedure, or specific help information for each command.
- trace – turns tracing facility on or off, in order to trace through the procedure execution.
- enable / disable – enables or disables the automatic expansion procedures on a specified database segment or device.
- who – shows any active expansion processes running currently. '<spid>' restricts the output for a particular spid.

If you provide no argument, sp_dbextend defaults to help. For more information on sp_dbextend, see the *Reference Manual*.

Note The automatic expansion procedure does not create new devices; it only alters the size of the database and segment on existing devices to which the segment currently maps.

Dropping the threshold action procedure

To discontinue threshold action procedure, clear the threshold using the sp_dropthreshold, or use sp_dbextend with the *clear* option. For information about sp_dropthreshold, see the *Reference Manual*.

Testing the process with *sp_dbextend*

You can use the check parameter to validate the current settings of various thresholds. For instance, check warns you if multiple segments share the same set of devices and both segments are set for automatic expansion, or if the threshold currently set to trigger automatic expansion on the logsegment is too close to the current last-chance threshold for the logsegment. In this situation, the automatic threshold does not fire, and check reports a warning.

sp_dbextend offers a powerful simulation mode that any user with sa_role permission can use to simulate the execution of the top-level threshold action procedure.

- To define expansion policies for the logsegment in the pubs2 database:

```
sp_dbextend 'set', 'database', pubs2, logsegment,'3M'

sp_dbextend 'set', 'threshold', pubs2, logsegment, '1M'
```

- To simulate expansion for these policies:

```
sp_dbextend 'simulate', pubs2, logsegment
------------------------------
```

Messages from the server follow this input.

The following outputs show the series of database and disk expansions that would occur if the threshold on database pubs2 segment logsegment fired once:

```
sp_dbextend 'simulate', pubs2, logsegment
---------------
NO REAL WORK WILL BE DONE.
Simulate database / device expansion in a dry-run mode 1 time(s).
These are the series of database/device expansions that would have
happened if the threshold on database'pubs2',
segment 'logsegment' were to fire 1 time(s).

Threshold fires: Iteration: 1.
============================

Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'.

Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M'
-- Segment: logsegment

Database 'pubs2' was altered by total size '3M' for
```

```
segment 'logsegment'.
Summary of device/database sizes after 1 simulated extensions:
================================================
devicename initial size final size
---------- ------------ ----------
pubs2_data 20.0M        20.0M

(1 row affected)

Database 'pubs2', segment 'logsegment' would be altered from an initial
size of '4M' by '3M' for a resultant total size of '7M'.

To actually expand the database manually for this threshold, issue:
sp_dbextend 'execute', 'pubs2','logsegment', '1'

(return status = 0)
```

To expand the database manually for this threshold, execute:

```
sp_dbextend 'execute', 'pubs2', 'logsegment'
--------------
```

This output shows that if the threshold fires at this level, an alter database command operates on the pubs2_data device for the logsegment:

```
sp_dbextend 'execute', pubs2, logsegment

Threshold fires: Iteration: 1.
================================
Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'. Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M' -- Segment: logsegment
Extending database by 1536 pages (3.0 megabytes) on disk pubs2_data
Warning: The database 'pubs2' is using an unsafe virtual device
'pubs2_data'. The recovery of this database can not be guaranteed.

Warning: Using ALTER DATABASE to extend the log segment will cause user
thresholds on the log segment within 128 pages of the last chance
threshold to be disabled.

Database 'pubs2' was altered by total size '3M' for segment 'logsegment'.

(return status = 0)
```

- To simulate what would actually happen if the threshold fired <*n*> times in succession on a particular segment, issue the same command, specifying the number of iterations:

```
sp_dbextend 'simulate', pubs2, logsegment, 5
-----------------
```

The following example shows how to expand this database five times:

```
sp_dbextend 'execute', 'pubs2', 'logsegment', 5
------------------
```

The output this provides shows that firing the threshold five times in succession puts the database through a series of alter database operations, followed by one or more disk resize operations and, finally, an alter database on the specified device.

Setting up the pubs2 database for automatic expansion

To set up different segments in the pubs2 database for automatic expansion, follow this procedure. Not all these steps are mandatory. For example, you may choose not to set *growby* or *maxsize* for individual devices, and to use the system default policies only for the devices.

❖ **Setting up pubs2**

1 Create the database. Enter:

```
create database pubs2 on pubs2_data = "10m" log on pubs2_log = "5m"
```

2 Set the *growby* and *maxsize* policies for the pubs2_data device at 10MB and 512MB respectively. You can be in any database to set these policies; you need not be in the specified database. Enter:

```
exec sp_dbextend 'set', 'device', pubs2_data, '10m', '512m'
```

3 The system default *growby* policy is 10% for devices. Rather than set new policies for the pubs2_log device, you can modify this system default, choosing an appropriate *growby* value. The pubs2_log then expands at this rate. Enter:

```
exec sp_dbextend 'modify', 'device', 'default', 'growby', '3m'
```

4 Set the growby rate for the default segment, but do not specify a maximum size. Enter:

```
exec sp_dbextend 'set', 'database', pubs2, 'default', '5m'
```

The *growby* rate on the default segment may be different from that on the devices where the segment resides. *growby* controls the segment's expansion rate when it is running out of free space, and is used only when you expand the segment.

5 Set the *growby* and *maxsize* variables for the logsegment:

```
 exec sp_dbextend 'set', 'database', pubs2, 'logsegment', '4m', '100m'
```

6 Examine the policies established for various segments in the pubs2 database:

```
exec sp_dbextend 'list', 'database', pubs2
```

7 Examine the policies in the various devices that pubs2 spans. The pattern specifier for *devicename* ("%") picks up all these devices:

```
exec sp_dbextend 'list', 'device', "pubs2%"
```

8 Install the expansion threshold for the default and logsegments segments in pubs2. This sets up and enables the expansion process, and allows you to choose the free space threshold at which to trigger the expansion process. Enter:

```
 use pubs2
------------------------------------------------------------------

    exec sp_dbextend 'set', 'threshold', pubs2, 'default',   '4m'
    exec sp_dbextend   'set', 'threshold',  pubs2,  'logsegment', '3m'
```

9 Examine the thresholds installed by the commands above.

```
exec sp_dbextend list, 'threshold'

segment name free pages  free pages (KB) threshold procedure status
------------ ----------- --------------- ------------------- -----------
default          2048    4096             sp_dbxt_extend_db enabled
logsegment        160  320           sp_thresholdaction lastchance
logsegment       1536  3072          sp_dbxt_extend_db      enabled
Log segment free space currently is 2548 logical pages (5096K).
(1 row affected, return status = 0)
```

In this output, sp_dbxt_extend_db is the threshold procedure that drives the expansion process at runtime. The expansion thresholds are currently enabled on both the default and logsegment segments.

10 Use simulate to see the expansion:

```
exec sp_dbextend 'simulate', pubs2, logsegment
exec sp_dbextend 'simulate', pubs2, 'default', '5'
```

11 Use modify to change the policy if necessary:

```
exec sp_dbextend 'modify', 'database', pubs2, logsegment, 'growby','10m'
```

12 To disable expansion temporarily on a particular segment, use disable:

```
exec sp_dbextend 'disable', 'database', pubs2, logsegment
```

13 Examine the state of the expansion policy on databases and devices:

```
exec sp_dbextend list, 'database'
name         segment    item    value status
------------ ---------- ------- ----- --------
server-wide (n/a)      (n/a)   (n/a) enabled
default     (all)      growby  10%   enabled
pubs2       default    growby  5m    enabled
pubs2       logsegment growby  10m   disabled
pubs2       logsegment maxsize 100m  disabled
(1 row affected, return status = 0)
```

The status disabled indicates that the expansion process is currently disabled on the logsegment in pubs2.

```
exec sp_dbextend list, 'device'
name           segment item   value status
-------------- ------- ------ ----- -------
server-wide    (n/a)   (n/a)  (n/a) enabled
default        (n/a)   growby 3m    enabled
mypubs2_data_0 (n/a)   growby 10m   enabled
mypubs2_data_1 (n/a)   growby 100m  enabled
mypubs2_log_1  (n/a)   growby 20m   enabled
mypubs2_log_2  (n/a)   growby 30m   enabled
(1 row affected, return status = 0)
```

14 Use enable to reenable the expansion process:

```
exec sp_dbextend 'enable', 'database', pubs2, logsegment
```

Restrictions and limitations

- When threshold procedures are installed on multiple segments in one or more databases, the expansion is performed in the order in which the thresholds fire. If abort tran on log full is off for the logsegment, tasks wait until the threshold procedure for the logsegment is scheduled to alter the database.

- In unlogged segments, tasks continue to process even after the free space threshold is crossed, while the threshold procedure remains in queue. This can cause “out of space” errors in data segments. Design your thresholds to have sufficient space in the database for the task to complete.
- If many threshold procedures fire at the same time, the procedure cache may become overloaded. This is more likely to occur in installations with large numbers of databases, many segments, and many threshold action procedures installed.
- If the space in the tempdb is very low, and other operations need tempdb resources, the threshold procedures may fail even while trying to correct the situation. Make sure that threshold procedures in tempdb are installed with sufficiently large amounts of free space, at least 2MB, to avoid this problem.

You may need to change your dump and load procedures to manage site-specific policies that determine how databases and devices are expanded.

Dumping a database does not transport information stored in master.db.sysattributes, so if you use dump and load as a way to migrate databases from a source server to a target server, you must manually migrate any site-specific policies encoded as data in the sysattributes database. There are two possible workarounds:

- Using bcp out from a view defined on master.dbo.sysattributes for entries with class number 19, you can manually extract the data from master.dbo.sysattributes, then use bcp in to load the data into the target server. This requires that both databases across the two servers have the same segment IDs.
- You can also use the ddlgen feature of Sybase Central to regenerate the sp_dbextend set invocations necessary to re-create your policy rules, by running the ddlgen script at the target server. However, renamed logical devices across servers cannot be managed by the ddlgen procedure. You must rename the devices manually at the target server.

The following restrictions do not cause failure.

- You can install a threshold action on an unlogged segment when the database has the option 'no free space acctg' turned on. This option means only that no database expansion is performed, since threshold actions are not fired with this option is off. Leaving this option on generates a warning message.

- Sybase recommends that you periodically dump the master database if expansion occurs, so that you can re-create the master database in case of failure after several expansions.
- Sybase recommends that you do not install these generic threshold procedures on any system databases, particularly the master database, as modifying space usage in the master database requires special treatment.
- You cannot use thresholds to shrink a database or segment.

CHAPTER 15

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 explains how to use thresholds to monitor the amount of free space in a database segment.

Monitoring free space with the last-chance threshold

All databases have a **last-chance threshold**, including master. The threshold is an estimate of the number of free log pages that are required to back up the transaction log. As you allocate more space to the log segment, Adaptive Server automatically adjusts the last-chance threshold.

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.)

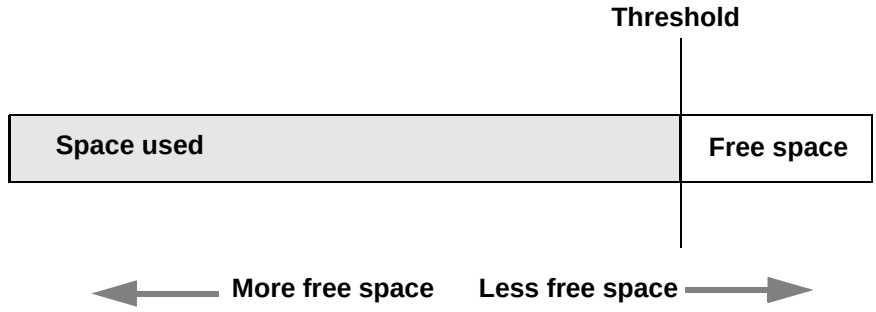
Figure 15-1 illustrates a log segment with a last-chance threshold. The shaded area represents log space that has already been used; the unshaded area represents free log space. The last-chance threshold has not yet been crossed.

Figure 15-1: 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 15-2: Executing sp_thresholdaction when the last-chance threshold is reached



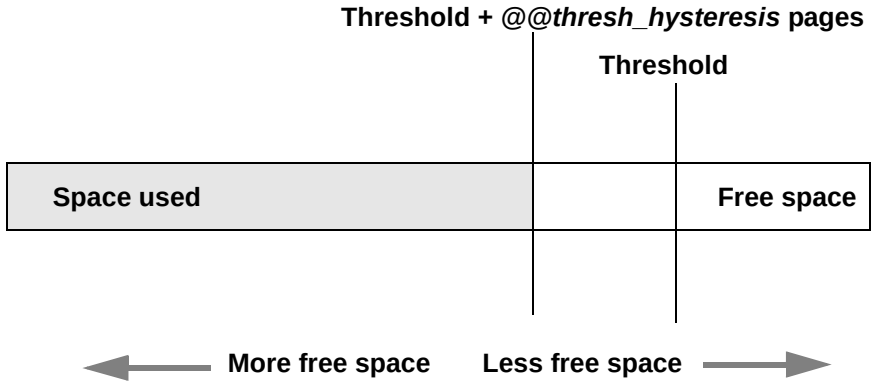
Controlling how often *sp_thresholdaction* executes

Adaptive Server uses a *hysteresis value*, the global variable *@@thresh_hysteresis*, to control how sensitive thresholds are to variations in free space.

A threshold is deactivated after it executes its procedure, and 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. You cannot change the value of *@@thresh_hysteresis*.

For example, when the threshold in Figure 15-2 executes sp_thresholdaction, it is deactivated. In Figure 15-3, the threshold is reactivated when the amount of free space increases by the value of *@@thresh_hysteresis*:

Figure 15-3: Free space must rise by @@thresh_hysteresis to reactivate threshold



Rollback records and the last-chance threshold

Adaptive Server version 11.9 and later include **rollback records** in the transaction logs. Rollback records are logged whenever a transaction is rolled back. Servers save enough space to log a rollback record for every update belonging to an open transaction. If a transaction completes successfully, no rollback records are logged, and the space reserved for them is released.

In long-running transactions, rollback records can reserve large amounts of space.

To check the space used by the syslogs, run sp_spaceused:

```
sp_spaceused syslogs
```

The output is:

```
name            total_pages     free_pages      used_pages      reserved_pages
--------------- --------------- --------------- --------------- --------------
syslogs               5632            1179            3783            670
```

The dbcc checktable(syslogs) produces similar output:

```
Checking syslogs: Logical pagesize is 2048 bytes
The total number of data pages in this table is 3761.
*** NOTICE: Space used on the log segment is 3783 pages (7.39 Mbytes), 67.17%.
*** NOTICE: Space reserved on the log segment is 670 pages (1.31 Mbytes), 11.90%.
*** NOTICE: Space free on the log segment is 1179 pages (2.30 Mbytes), 20.93%.
```

If the last chance threshold for the transaction log fires when it seems to have sufficient space, it may be the space reserved for rollbacks that is causing the problem. See "Determining the current space for rollback records" on page 463 for more information.

Calculating the space for rollback records

To calculate the increased amount of space to add to a transaction log to accommodate rollback records, estimate:

- The number of update records in the transaction log that are likely to belong to already rolled-back transactions
- The maximum number of update records in the transaction log that are likely to belong to open transactions at any one time

Update records are the records that change the timestamp value. They include changes to data pages, index pages, allocation pages, and so on.

Each rollback record requires approximately 60 bytes of space, or 3 one hundredths of a page. Thus, the calculation for including rollback records (RRs) in the transaction log is:

Added space, in pages = (logged RRs + # open updates) X 3/100.

You may also want to add log space to compensate for the effects of rollback records on the last-chance threshold and on user-defined thresholds, as described in the following sections.

Using lct_admin to determine the free log space

Use logsegment_freepages to determine the amount of free space your dedicated log segment has. The syntax is:

```
lct_admin("logsegment_freepages", database_id)
```

To see the number of free pages for the pubs2 database log segment:

```
select lct_admin("logsegment_freepages", 4)
---------------------
                     79
```

Determining the current space for rollback records

To determine the number of pages a database currently reserved for rollbacks, issue lct_admin with the reserved_for_rollbacks parameter. The partial syntax for lct_admin is:

```
select lct_admin ("reserved_for_rollbacks", dbid)
```

The number of pages returned are the number reserved, but not yet allocated, for rollback records.

For example, to determine the number of pages reserved for rollbacks in the pubs2 database (which has a dbid of 5), issue the following:

```
select lct_admin("reserved_for_rollbacks", 5)
```

See the *Reference Manual* for more information about lct_admin.

Effect of rollback records on the last-chance threshold

Adaptive Servers that use rollback records must reserve additional room for the last-chance threshold. The last-chance threshold (“LCT”) is also likely to be reached sooner because of the space used by already logged rollback records and the space reserved against open transactions for potential rollback records.

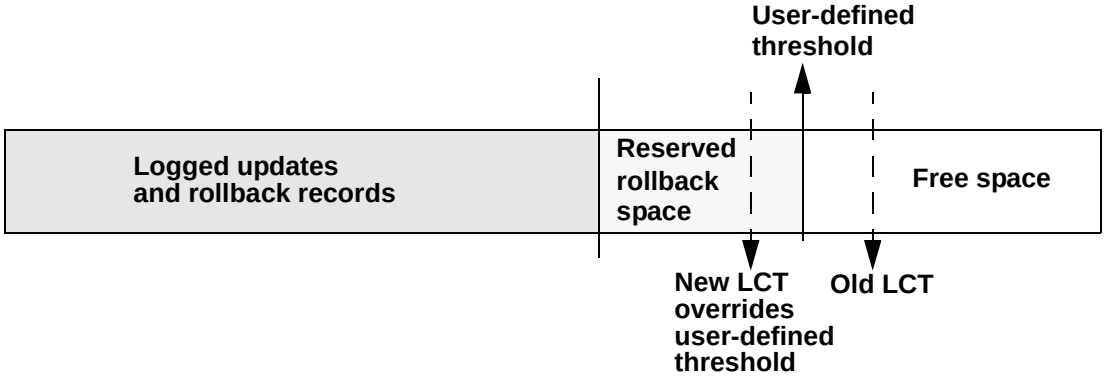
User-defined thresholds

Because rollback records occupy extra space in the transaction log, there is less free space after the user-defined threshold for completing a dump than in versions of Adaptive Server that do not use rollback records. However, the loss of space for a dump because of the increased last-chance threshold is likely to be more than compensated for by the space reserved for rollback records for open transactions.

A user-defined threshold is often used to initiate a dump transaction. The threshold is set so there is enough room to complete the dump before the last-chance threshold is reached and all open transactions in the log are suspended.

In databases that use mixed log and data, the last-chance threshold moves dynamically, and its value can be automatically configured to be less than the user-defined threshold. If this happens, the user-defined threshold is disabled, and the last chance threshold fires before the user-defined threshold is reached, as shown in Figure 15-4:

Figure 15-4: LCT firing before user-defined threshold



The user-defined threshold is reenabled if the value of last-chance threshold is configured to be greater than the user-defined threshold (for example, if the last chance threshold is reconfigured for the value of “Old LCT” in Figure 15-4).

In databases with a separate log segment, the log has a dedicated amount of space and the last-chance threshold is static. The user-defined threshold is not affected by the last-chance threshold.

Last-chance threshold and user log caches for shared log and data segments

Every database in an Adaptive Server has a last-chance threshold, and all databases allow transactions to be buffered in a user log cache. When you initially create a database with shared log and data segments, its last-chance threshold is based on the size of the model database. As soon as data is added and logging activity begins, the last-chance threshold is recalculated dynamically, based on available space and currently open transactions. The last-chance threshold of a database with separate log and data segments is based on the size of the log segment and does not vary dynamically.

To get the current last-chance threshold of any database, you can use lct_admin with the reserve parameter and a specification of 0 log pages:

```
select lct_admin("reserve",0)
```

The last-chance threshold for a database is stored in the systhresholds table and is also accessible through sp_helpthreshold. However, note that:

- sp_helpthreshold returns user-defined thresholds and other data, as well as an up-to-date value for the last-chance threshold. Using lct_admin is simpler if you need only the current last-chance threshold. Either of these values produce the most current value for the last-chance threshold.
- For a database with shared log and data segments, the last-chance threshold value in systhresholds may *not* be the current last-chance threshold value.

Reaching last-chance threshold suspends transactions

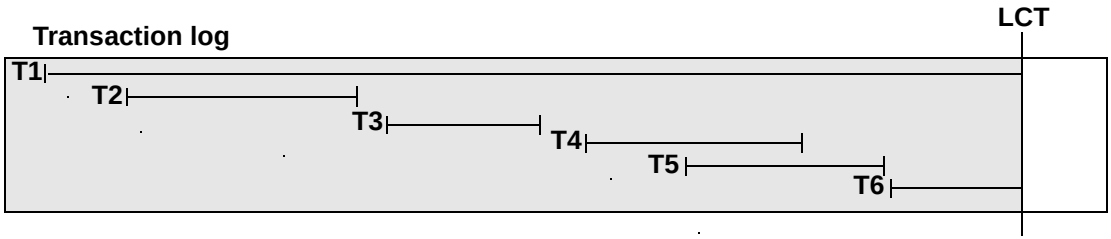
The default behavior of Adaptive Server is to suspend open transactions until additional log space is created. Transactions suspended because of the last-chance threshold can be terminated using the abort parameter of the lct_admin system function, described in "Using lct_admin abort to abort suspended transactions," below. For information about configuring Adaptive Server to automatically abort suspended processes, see "Automatically aborting or suspending processes" on page 468.

Using *lct_admin abort* to abort suspended transactions

All open transactions are suspended when the transaction log reaches the last-chance threshold. Typically, space is created by dumping the transaction log, since this removes committed transactions from the beginning of the log. However, if one or more transactions at the beginning of the log is still open, it prevents a dump of the transaction log.

Use lct_admin abort to terminate suspended transactions that are preventing a transaction log dump. Since terminating a transaction closes it, this allows the dump to proceed. Figure 15-5 illustrates a possible scenario for using lct_admin abort:

Figure 15-5: Example of when to use of lct_admin abort



In Figure 15-5, a transaction log has reached its LCT, and open transactions T1 and T6 are suspended. Because T1 is at the beginning of the log, it prevents a dump from removing closed transactions T2 through T5 and creating space for continued logging. Terminating T1 with lct_admin abort allows you to close T1 so that a dump can clear transactions T1 through T5 from the log.

lct_admin abort replaces lct_admin unsuspend.

lct_admin abort syntax

The syntax for lct_admin abort is:

```
lct_admin("abort", {process_id [, database_id]})
```

Before you can abort a transaction, you must first determine its ID. See "Getting the process ID for the oldest open transaction," below for information about determining the transaction's process ID.

To terminate the oldest transaction, enter the process ID (spid) of the process that initiated the transaction. This also terminates any other suspended transactions in the log that belong to the specified process.

For example, if process 83 holds the oldest open transaction in a suspended log, and you want to terminate the transaction, enter:

```
select lct_admin("abort", 83)
```

This also terminates any other open transactions belonging to process 83 in the same transaction log.

To terminate all open transactions in the log, enter:

```
select lct_admin("abort", 0, 12)
```

Getting the process ID for the oldest open transaction

Use the following query to find the spid of the oldest open transaction in a transaction log that has reached its last-chance threshold:

```
use master
go
select dbid, spid from syslogshold
where dbid = db_id("name_of_database")
```

For example, to find the oldest running transaction on the pubs2 database:

```
select dbid, spid from syslogshold
where dbid = db_id ("pubs2")
dbid   spid
------ ------
     7      1
```

Using *alter database* when the master database reaches the last-chance threshold

When the last-chance threshold on the master database is reached, you can use alter database to add space to the master database's transaction log. This allows more activity in the server by causing suspended transactions in the log to become active. However, while the master transaction log is at its last-chance threshold, you cannot use alter database to make changes in other databases. Thus, if both master and another database reach their last-chance thresholds, you must first use alter database to add log space to the master database, and then use it again to add log space to the second database.

Automatically 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 or readtext) and commands that might be necessary to free additional log space (dump transaction, dump database, and alter database) can be executed.

Using *abort tran on log full* to abort transactions

To configure the last-chance threshold to automatically abort open transactions, rather than suspend them, enter:

```
sp_dboption database_name "abort tran on log full", true
```

If you upgrade from an earlier version of Adaptive Server, the newly upgraded server retains the abort tran on log full setting.

Waking suspended processes

After dump transaction 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, you can run dump tran with truncate_only, which runs even in a situation with nonlogged-writes. If log space is so critical that dump tran with truncate_only fails, you can run dump tran with no_log. However, dump tran with no_log is for emergencies only, and should be run only as a last resort.

After the transaction dump completes, and transactions have successfully been freed from the log suspend state, the system administrator or database owner can dump the database.

If this does not free enough space to awaken the suspended processes, increase the size of the transaction log. Use the log on option of alter database to allocate additional log space.

Short of killing the command, one can also simply abort it, with the Transact-SQL builtin lct_admin("abort", <spid>). Aborting the command might be preferable to killing the connection. That builtin can only be applied to a process that is sleeping in log suspend.

If you have system administrator priviledges, you can use the sp_who command to determine which processes are in a log suspend status, then use the kill command to waken the sleeping process. To avoid killing processes, you can use lct_admin("abort", *spid*) to abort the process. You can use lct_admin only on processes that are sleeping in log suspend.

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.

sp_addthreshold, sp_modifythreshold, and sp_dropthreshold 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 sp_helpthreshold to get information about all thresholds in a database. Use sp_helpthreshold *segment_name* to get 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, you must enclose it 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 a free-space threshold instead of a last-chance 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. 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 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 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 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 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.

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 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 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 477.

Testing and adjusting the new threshold

Use dump transaction to make sure your transaction log is less than 55 percent full. Then use the following procedure to test the new threshold:

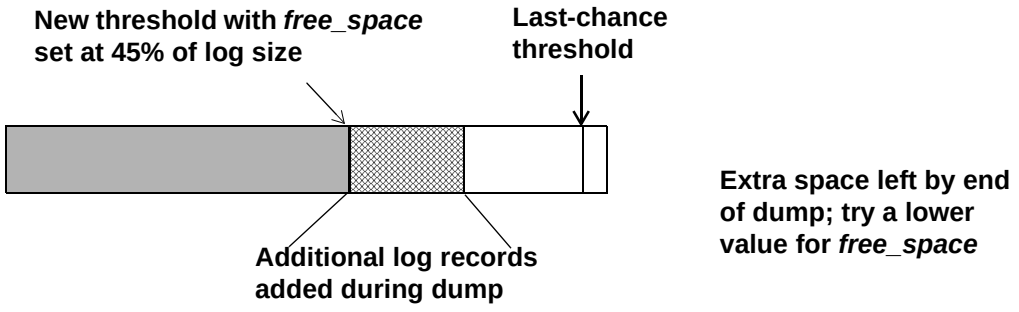
1 Fill the transaction log by simulating routine user action. Use automated scripts that perform typical transactions at the projected rate.

When the 45 percent free-space threshold is crossed, your threshold procedure dumps the transaction log. Since this is not a last-chance threshold, transactions are not suspended or aborted; the log will continues 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.

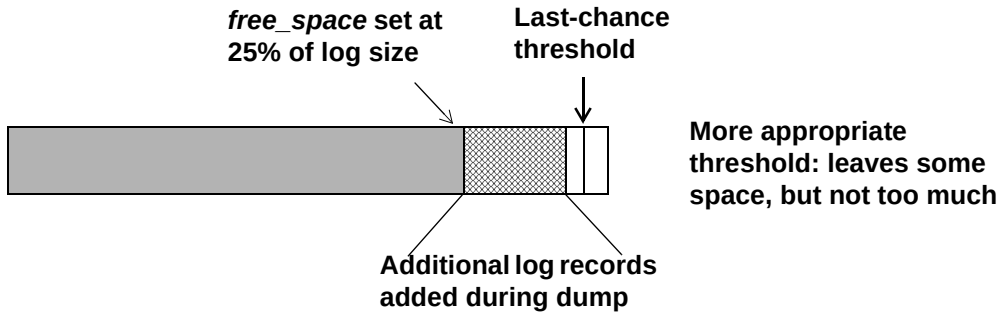
3 If considerable space was left in the log when the dump completed, you may not need to dump the transaction log so soon, as shown in Figure 15-6:

Figure 15-6: Transaction log with additional threshold at 45 percent



Try waiting until only 25 percent of log space remains, as shown in Figure 15-7:

Figure 15-7: Moving threshold leaves less free space after dump

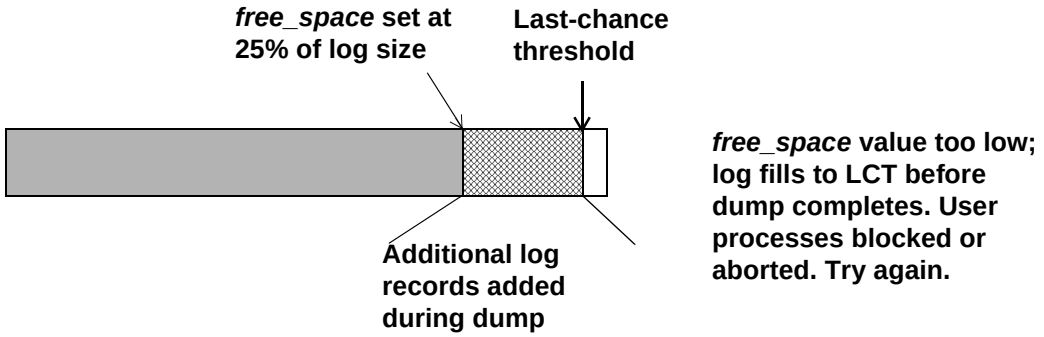


Use sp_modifythreshold to adjust the *free_space* value to 25 percent of the log size. For example:

```
sp_modifythreshold mydb, logsegment, 512,
    thresh_proc
```

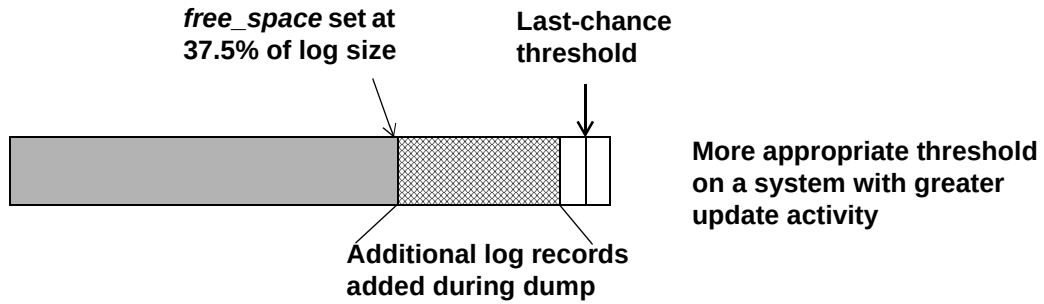
4 Dump the transaction log and test the new *free_space* value. If the last-chance threshold is crossed before the dump completes, you are not beginning the dump transaction soon enough, as shown in Figure 15-8:

Figure 15-8: 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 15-9:

Figure 15-9: 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 before your users encounter problems.

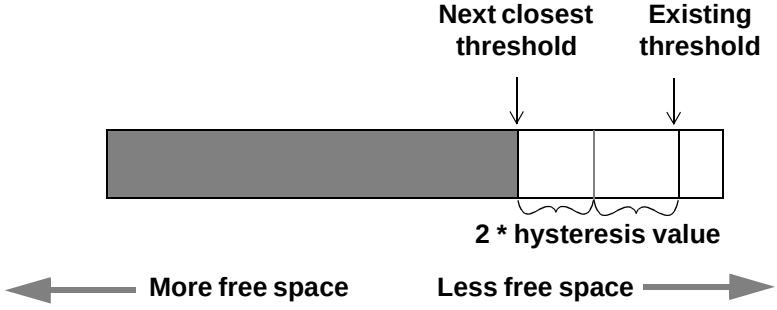
The following example creates a free-space threshold on the default segment of mydb. When the free space on this segment falls below 200 pages, Adaptive Server executes the procedure space_dataseg:

```
sp_addthreshold mydb, "default", 200, space_dataseg
```

Determining threshold placement

Each new threshold must be at least twice the *@@thresh_hysteresis* value from the next closest threshold, as shown in Figure 15-10:

Figure 15-10: Determining where to place a threshold



To see the hysteresis value for a database, use:

```
select @@thresh_hysteresis
```

In this example, a segment has a threshold set at 100 pages, and 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 mydb, user_log_dev, 228,
    sp_thresholdaction
```

Creating threshold procedures

Sybase does not supply threshold procedures. You must create these procedures yourself to ensure that they are tailored to your site's needs.

Suggested actions for a threshold procedure include writing to the server's error log and dumping the transaction log to increase the amount of log space. You can also execute remote procedure calls to an Open Server or to XP Server. For example, if you include the following command in sp_thresholdaction, it executes the procedure mail_me on an Open Server:

```
exec openserv...mail_me @dbname, @segment
```

See the *Transact-SQL User's Guide* for more information on using extended stored procedures and XP Server.

This section provides some guidelines for writing threshold procedures, as well as 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

Include a print statement near the beginning of your procedure to record the database name, segment name, and threshold size in the error log. If your procedure does not contain a print or raiserror statement, the error log will not contain any record of the threshold event.

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. dump transaction 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.

When there is enough log space, suspended transactions are awakened. If you abort transactions rather than suspending them, users must resubmit them.

If sp_thresholdaction failed because of non-logged-writes status, you can issue dump tran with no_log as an alternative.

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 must copy the files back to the disk to reload them.)

Your choice depends 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, which can be any of:

- 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 unlogged 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; threshold procedures on your data segments do not execute when these holds are crossed. 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! If you disable free-space accounting, system procedures cannot provide accurate information about space allocation.

After you disable data segment free-space accounting, the counts may be inaccurate, even after you set no free space acctg to false. To force Adaptive Server to recalculate, issue shutdown with nowait and then restart the Adaptive Server. This may increase recovery times.

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W