SYBASE®

Migrating to Sybase® Adaptive Server™ Enterprise 11.5

Adaptive Server™

Document ID: 34982-01-1150-02

December 1997

Copyright Information

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

Table of Contents

4. Prepare: Writing a Plan and Getting Ready to Migrate

5. Implement: Making Required Application Changes

6. Implement: Making Required Database Administration Changes

7. Test: Ensuring Stability and Performance

A. The Cyrano Migration Pack

B. Worksheets for Your Current Environment

C. Sample Migration Plan Outline

D. Sample Migration Task Lists

E. Migration Plan Checklist

List of Tables

1 Introduction and Guide to Resources

In This Chapter

This chapter gives an overview of the topics covered in this manual. In addition, it points you to an extensive list of Sybase resources that can help you plan and execute a trouble-free migration to Sybase Adaptive Server Enterprise(TM) 11.5.

This chapter covers the following topics:

In This Manual

This manual is based on SAFE / EM migration methods. SAFE / EM, a methodology for migration developed by Sybase Professional Services, is organized into the following phases:

- **Analyze**
- **Prepare**
- **Implement**
- **Test**

The chapters in this manual are:

Table 1-1: Chapters in this migration guide

Chapter Number	Chapter Title	Contents
1	"Introduction and Guide to Resources"	A survey of resources available from Sybase and third-party vendors.
2	"Analyze: Documenting Business Requirements"	A series of worksheets for documenting the flow of information in your system and your business requirements.
3	"Analyze: Documenting Your Environment"	A series of worksheets for documenting your environment, including hardware and software, for compatibility assessment.
4	"Prepare: Writing a Plan and Getting Ready to Migrate"	Guidelines for determining the best migration plan for your site.
5	"Implement: Making Required Application Changes"	A technical summary T-SQL syntax, query processing, reserved word and system changes that can affect your applications and cause failure and unexpected results.
6	"Implement: Making Required Database Administration Changes"	A technical summary system changes that can require you to make adjustments to your database administration procedures.
7	"Test: Ensuring Stability and Performance"	Guidelines for setting up a test environment and writing test suites.

Relating Documentation to Migration Phase

Sybase provides documentation for all stages of the migration process. While this migration guide documents the minimum changes to your system and applications necessary to avoid problems, we recommend that you refer to *What's New in Sybase Adaptive Server Enterprise 11.5?* and other Sybase manuals to help you plan the design of your new Adaptive Server system to take advantage of Sybase's new performance features.

Table 1-2 gives general guidelines for relating migration phase to Sybase documentation.

Table 1-2: Documentation for Migration

Document	Time period covered	Range of tasks covered
• What's New in Adaptive Server Enterprise 11.5 • Migrating to Sybase Adaptive Server Enterprise 11.5	Planning / preparation prior to upgrade	Assessing current system Planning migration Making applications compatible Updating DBA procedures
• Release Bulletins, TechNotes (http: / / techinfo.sybase.com)	Planning / preparation prior to upgrade	Finding information needed to avoid upgrade problems, including problem reports, special installation issues, and compatibility issues
• Installing Sybase Products	Upgrade preparation and implementation	Preparing system to upgrade Installing software Performing upgrade tasks
• System Administration Guide • Performance and Tuning Guide	Planning / preparation prior to migration and testing after upgrade	Planning system design for Adaptive Server 11.5 Monitoring and tuning system for increased performance

SyBooks

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

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

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

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

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

Technical Information Library

Technical Information Library is a collection of technical documents, such as TechNotes, FAQs, and white papers, produced and used by Sybase engineers to troubleshoot customer issues and enhance the use of Sybase products. Technical Information Library also contains information for downloading EBFs from the Web. For information on certifications and EBFs, follow the steps below.

For the latest information on product certifications and/or the EBF Rollups:

1. Point your Web browser to the Technical Information Library at the following URL:

   http://techinfo.sybase.com

2. In the Browse section, click on the Hot entry.
3. Explore your area of interest: Hot Docs covering various topics, or Hot Links to Technical News, Certification Reports, Partner Certifications, and so on.

If you are a registered Support*Plus* user:

1. Point your Web browser to the Technical Information Library at the following URL.

   http://techinfo.sybase.com

2. In the Browse section, click on the Hot entry.
3. Click on the EBF Rollups entry.
4. Follow the instructions associated with the Support*Plus*SM Online Services entries.

   You can research EBFs using the Technical Information Library, and you can download EBFs using Electronic Software Distribution (ESD).

If you are not a registered Support*Plus* user, and you want to become one:

You can register by following the instructions on the Web.

To use Support*Plus*, you need:

- A Web browser that supports the Secure Sockets Layer (SSL), such as Netscape Navigator 1.2 or later
- An active support license
- A named technical support contact
- Your user ID and password

➤ ***Note***

You can reach Support *Plus* Online Services by going to the Sybase home page at http://www.sybase.com and choosing “Technical Support”. Current EBF’s can be downloaded by registered Support *Plus* users.

Migration Web Page

For an overview of Sybase migration products and services, as well as special programs, see the Sybase Migration Resource Guide at http://www.sybase.com/migration. Click the link to SAFE/EM (Sybase Advanced Framework to Enable Effective Migration) for information on migration consulting from Sybase Professional Services.

Migration Courses

Sybase Educational Products and Technology offers several courses to help you move to Adaptive Server 11.5. For information on courses and educational products, call 1-800-8-SYBASE or your local Sybase vendor.

- Technical Overview: Adaptive Server Enterprise 11.5 (1 day)

  Available at Sybase and partner learning centers, by arrangement at your site, or on CD-ROM.

- Powering Up with Adaptive Server Enterprise 11.5 (3 days)

  Available at Sybase and partner learning centers, by arrangement at your site, or on video. The Overview CD-ROM is included.

- Tools and Methods for Migration (2 days)

  The 11.5 version of this course will be available in November 1997. Can be taught at your site, downloaded from the Sybase

Migration Resource Guide on the Web
(http://www.sybase.com/migration), or ordered on CD-ROM.

Sybase Press Books

Sybase Press produces commercial books about Sybase topics, including *Upgrading and Migrating to Sybase SQL Server 11*, by Mitchell Gurspan. This book contains valuable information about the migration process, using operating system tools to aid your migration, and technical issues up to SQL Server 11.0.

To order this or other Sybase Press books, contact the online bookstore, AbyssBooks, by one of these methods:

- On the Web at
  http://www.abyssbooks.com/sybasestudent.html
- By email at orders@abyssbooks.com
- By phone toll-free at 1-888-WWW-Book

Third-Party Books

The following books provide background to the ANSI SQL-related changes to Transact-SQL which were made after version 4.9.2.

- Understanding the new SQL: A Complete Guide, by Jim Melton and Alan R. Simon
  (The Morgan Kaufmann Series in Data Management Systems)
  Paperback, 536 pages
  Published by Morgan Kaufman Publishers
  Publication date: March 1993
  ISBN: 1558602453
- The SQL Standard (fourth edition)
  by Hugh Darwen, C. J.
  4th Edition
  Paperback, 572 pages
  Published by Addison-Wesley Pub Co
  Publication date: April 1997
  ISBN: 0201964260

Consulting Services

Sybase Professional Services offers consulting services to customers planning to migrate to Adaptive Server 11.5. Professional Services

uses the Sybase-developed and recommended methodology, SAFE/EM (Sybase Advanced Framework to Enable Effective Migration). The migration technology in this manual is based on SAFE/EM.

For new or upgrading customers, Professional Services offers SAFE/ITA (Sybase Advanced Framework to Enable Information Technology Architecture), an architectural framework and ongoing process for making high-value IT investments that support your business goals.

Third-Party Tools

Sybase recommends the **Cyrano Migration Pack** to facilitate the handling of technical issues related to migration and monitor the performance of your 11.5 installation. For more information about Cyrano, see Appendix A, "The Cyrano Migration Pack" and Cyrano's Web site at http://www.cyrano.com.

Other third party tools which may be useful include:

- Cast workbench (http://castsoftware.com)
- Embarcadero DBArtisan (http://embarcadero.com)
- Platinum Desktop DBA (http://www.platinum.com/products/sybase.htm)
- SFI SQL Programmer (www.sfi-software.com)
- SQL BackTrack (http://www.datatools.com)

If You Need Help

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

2 Analyze: Documenting Business Requirements

In This Chapter

In this chapter you begin the first phase of migration planning: documenting your environment. This chapter helps you organize business information required for an effective migration plan. The sections are:

See Appendix B, “Worksheets for Your Current Environment,” for worksheets like the ones used in the examples in this chapter.

Diagramming the System

Create a diagram (or table or written description) illustrating the flow of information on your system. This will serve as a reference for the migration team. Include the following information:

- Servers, including file, print and application servers
  - Machine name
  - IP address
  - Sybase name and aliases
- Clients
  - Applications
  - Number of users
- Network
  - Protocols
  - Gateways
  - Routers, brouters, bridges

For example, you can create a diagram like the one in *Figure 2-1: High-level system diagram*, a high-level view of brokerage firm with business in several cities and two main computing centers.

Acme Brokerage Company

System Architecture

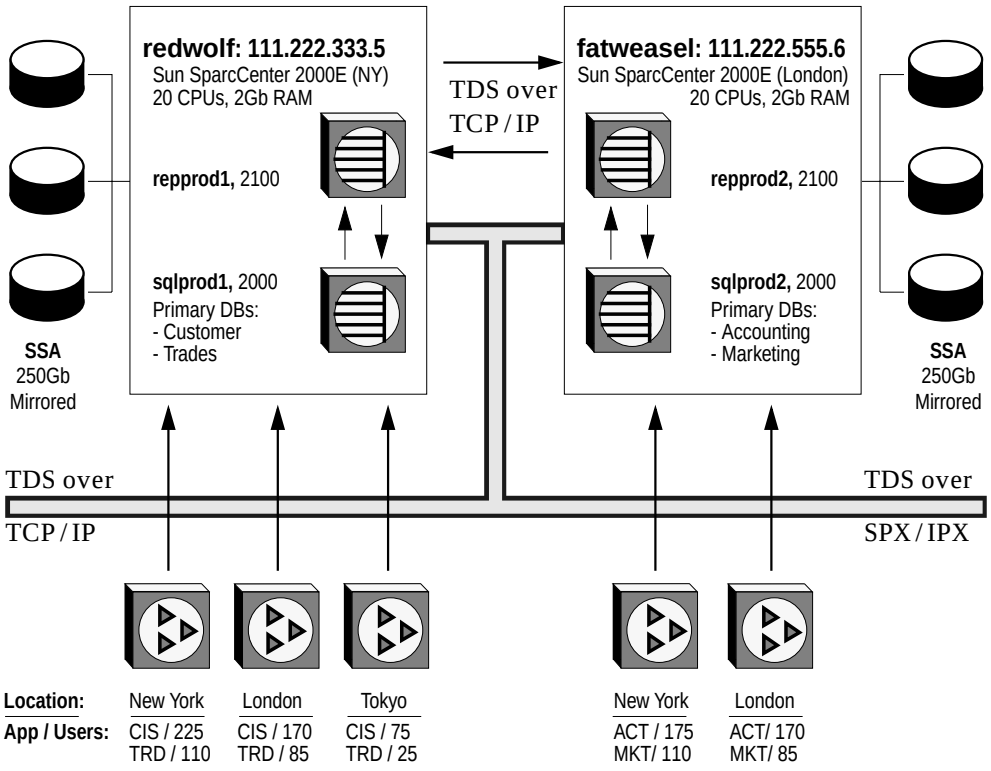

Figure 2-1: High-level system diagram

You can write a high-level business description in addition to or instead of a diagram.

Operational Business Requirements

This section suggests ways to document your operational business requirements. You will use these baseline requirements to help you plan the migration and develop success criteria. This section covers the following issues:

Availability Requirements

Record the times users need to access databases, as well as maximum acceptable downtime, as in the following example:

Table 2-1: Availability Requirements

Database Name	Operational Hours	Maximum Downtime	General Comments
TRD	07:00 – 23:00 Mon to Fri	5 min	
CIS	07:00 – 23:00 Mon to Fri	15 min	
ACT	07:00 – 21:00 Mon to Sat	5 min	
MKT	07:00 – 20:00 Mon to Fri	30 min	

Database Change Metrics

For all databases, record:

- Database size
- Transaction log growth

- Table rowcounts and daily change rate (number of inserts, deletes, and updates.

Database Dump Details

Document your dump procedures, including times and devices, as in the following example:

Table 2-2: Database Dump Details

Database Name	Frequency of Database Dumps	Dump Device Used	Frequency of Transaction Log Dumps	Dump Device Used	Comments
master	every night	master_dumpdev			
TRD	every night	TRD_tape1 TRD_tape2	Every 15 min.	TRD_tape2	
CIS	every night	CIS_tape1	Every 15 min	CIS_tape3	
ACT	every night	ACT_tape1 ACT_tape2	Every 15 min	ACT_tape3	

Maintenance Procedures

Document your schedule for running data consistency checking and performance monitoring tools. Use a table similar to this example:

Table 2-3: Maintenance Procedures

Database Name	Frequency of *dbcc checkdb* and *dbcc checktable*	Frequency of *dbcc checkalloc* and *dbcc tablealloc*	Frequency of *update statistics*	Frequency of Monitoring Utilization
master	every night			

Table 2-3: Maintenance Procedures

Database Name	Frequency of *dbcc checkdb* and *dbcc checktable*	Frequency of *dbcc checkalloc* and *dbcc tablealloc*	Frequency of *update statistics*	Frequency of Monitoring Utilization
TRD	every weekend	different tables "Round-robin" every night	different tables "Round-robin" every night	every hour
CIS	every weekend	different tables "Round-robin" every night	different tables "Round-robin" every night	every hour
ACT	every weekend	different tables "Round-robin" every night	different tables "Round-robin" every night	every hour

Service Level Requirements

Document application details and service requirements, as in this example:

Table 2-4: Service Level Requirements

Application Name	Type of application	App. Language	Client Machines	No. of Con-current Users	Databases Accessed	Availability Require-ments (Per day)	Performanc e (Average Response Time)
Trades	Heavy OLTP	C	UNIX work-stations	220	TRD CIS	<5 min downtime	<2 sec
Customer	Light OLTP DSS	Power-builder	PC	470	CIS	<10 min downtime	< 5 sec
Accounting	Light OLTP DSS Batch	Power-builder	PC	345	ACT CIS	<5 min downtime	<5 sec
Marketing	DSS	C	PC	195	MKT CIS	<30 min downtime	<120 sec

Transaction Profile

Using **statistics io**, **showplan**, and both **dbcc 302** and **dbcc 310** to capture application processing details, record transaction profiles on a chart like the following. Save **showplan** and **dbcc** output for all critical transactions. You will use this information as a baseline for post-upgrade testing, as described in Chapter 7, "Test: Ensuring Stability and Performance."

See the *Performance and Tuning Guide* for more information on gathering transaction statistics. Also see "Resolving Optimizer Problems / Symptoms," in Technical Information Library.

Table 2-5: Transaction Profile

App Name	Process (Xact) Name	Type of Process-ing	Xact Prior-ity	Freq. per User (per hour)	Source Code	Required Average Response Time	Required Maximum Response Time	Current Avg/Max Response Time
Trades	addTrade	Heavy OLTP	P1	90	Stored Proc	<2 sec	<5 sec	1 sec avg 3 sec max
Trades	bustTrade	Heavy OLTP	P1	10	Stored Proc	<5sec	<10 sec	2 sec avg 8 sec Max
Trades	reconcileTrades	Batch	P1	1 per day	Embed-ded SQL / CO BOL	<30 min	<60 min	25 min avg 45 min max
Trades	listAccounts	Light OLTP	P1	180	Stored Proc	<2 sec	<5 sec	1 sec avg 2 sec max

Current Performance Metrics

Record as much of the following performance information as possible. Sybase monitors include SQL Monitor Client, Historical Monitor, and **sp_sysmon**.

- CPU utilization
  - Use operating system and Sybase monitors to measure the average and maximum CPU utilization (aggregate and per CPU on SMP servers) per "time window" (for example, online or batch) per server.
- Disk I / O

  - Use operating system monitors to measure I/Os per second per disk and controller, and I/O queue lengths per "time window" per server.
  - Use Sybase monitors to measure total I/Os, reads, and writes per second per Sybase device per "time window" per server.
- Concurrency
  - Use Sybase monitors to determine the average lock contention. Network I/O
  - Use Sybase monitors (Monitor Client or Historical Monitor) to record average execution rates for critical stored procedures.
  - Use operating system monitors to measure the packets per second per network interface card per "time window" per server.
  - Use Sybase monitors to measure the TDS packets ("sent from" and "received by") per "time window" per server.
- Memory
  - Use operating system monitors to determine the paging/swapping rates per second per "time window" per server.
  - Use Sybase monitors to determine the data and procedure cache hit rates per "time window" per server.

See the *Performance and Tuning Guide* for information about using Sybase tools and stored procedures to monitor performance factors.

Additional Business Requirements

Document any other important operational business requirements, such as:

- Priority list of applications to be migrated
- Constraints
  - Need to avoid year/quarter end processing?
  - How much downtime is ok?
  - Should the upgrade take place over the weekend
  - What staff is available: DBAs, system administrators?
  - What resources are available: Replication Servers, machines, tools such as Cyrano Migration Pack, funding?

- Application and data server dependencies
  - Is more than one application using the same SQL Server?
  - Can all applications on a server be migrated
- Vendor issues

3 Analyze: Documenting Your Environment

This chapter provides guidelines for documenting system hardware and software in the SQL Server production environment. This information is used to identify resource issues during the planning phase of the migration.

See Appendix B, “Worksheets for Your Current Environment” for worksheets like the ones used in the examples in this chapter.

In This Chapter

This chapter explains how document:

Hardware Configuration

Document your hardware environment as described in the following sections:

General Server Hardware

List the following hardware information for every server machine:

- Make and model
- Your customer ID with the vendor
- Technical support information

- Telephone number
- Support hours
- Name of your account manager and his or her telephone or pager number
- Web page if applicable

CPU Resources per Machine

List the following CPU information for each server machine:

- Total number of processors and their speed
- Number of processors available to Sybase SQL Server
- Other CPU-intensive processes sharing those processors
- List of processes / threads bound to specific CPUs
- List of processes / threads run at high priority

Disk Configuration

Use worksheets like those in the following examples to gather the following disk I / O information:

- Controller map
- Disk layout map
- Disk partition map
- Logical volume map

Table 3-1: Controller map

Controller Number	Make and Model	Firmware Revision	Months in Service	Transfer Rate (Kb/sec)
0	Sun Sparc 5	1.50	9	7500
1	Sun Sparc 20	1.00	6	10000

Table 3-2: Disk layout map

Physical Device name	Make and Model	Firmware Revision	Months in Service	Controller Number	Capacity (Mb)	Through-put (I/Os per sec)	Transfer Rate (Kb/sec)
c0t0d0	Seagate ST43401N	2.15	9	0	2900	80	1500
c0t0d1	Seagate ST43401N	2.15	12	0	2900	80	1000
c0t0d2	Seagate ST43401N	2.15	24	0	2900	80	1600
c0t0d3	Seagate ST43401N	2.00	16	0	2900	80	1200

Table 3-3: Logical volume map

Logical Volume Name	Member Disk Partitions	Used by (Sybase, UFS)	Sybase Device Name	Mirrored Logical Device	Capacity (Mb)	Stripe Width (Mb)
lv dev1	c0t0d0s3 c0t0d0s4 c0t0d1s3 c0t0d1s4	sybase	TRD Log	lv dev1 mirror	500	4
lv dev2	c0t0d0s3 c0t0d0s4 c0t0d1s3 c0t0d1s4	sybase	CIS Log	lv dev2 mirror	500	4

Table 3-4: Disk partition map

Physical Device Name	Partition Number	Used by (Sybase, UFS)	Device Name	OS Mirror Device Name	Capacity (mb)	Cylinder Range
c0t0d0	s0	disk label			2	0 – 1
	s2	backup				
	s3	swap	swap		998	2 – 501
	s4	sybase	TRD Log	c1t0s4	500	502 – 752
	s5	sybase	CIS Log	c1t0s5	500	753– 1003
	s6	ufs	/usr		900	1004 – 2733
c0t0d2	s0	disk label			2	0 – 1
	s2	backup				
	s3	swap	swap		2900	2 – 2733

Network Configuration

Use a worksheet like the one in the following example to show network interface card information for server and client machines:

Table 3-5: Network interface card layout map

Physical Device Name	Make and Model	Firmware Revision	Months in Service	Protocols supported	Network Address	Transfer Rate (Kb/sec)
c0t0d0	Sun Sparc 5	1.5	9	TCP/IP SPX/IPX	121.222.233.11	7500
c0t0d1	Sun Sparc 20	1.00	12	TCP/IP SPX/IPX	121.222.555.33	7500

Tape Configuration

Use a worksheet like the one in the following example to record tape or other storage media configuration:

Table 3-6: Tape layout map

Physical Device Name	Make and Model	Firmware Revision	Months in Service	Controller Number	Capacity (Mb_)	Transfer Rate (Kb/sec)
/dev/rmt/0	Sun Sparc 10	2.15	9	2	2000	500
/dev/rmt/1	Sun Sparc 20	1.00	12	2	2900	500

Physical Memory Utilization

List all the major processes running on a server machine and use the formulas given here to calculate their memory requirements. Add the individual sizes together for total memory requirements.

Use this table as a guide:

Table 3-7: Server memory map

Name	Runtime Memory Usage Calculation
Operating System	OS-specific
Networking Software	OS-specific
SQL Server	kernel only
SQL Server	total memory + additional netmem + extent i/o buffers (configuration parameters)
Backup Server (Local)	binary + (110Kb * number of stripes)
Backup Server (Remote)	binary + (54Kb * number of stripes)
sybmultbuf	binary only (attaches to BS shared memory)
Replication Server	binary + 7Mb + (500Kb * (number of DSI connections + number of LTM connections)

Table 3-7: Server memory map

Name	Runtime Memory Usage Calculation
Monitor Server	binary only
Gateways {List:}	
Other Applications {List:}	
Total Memory Required	
Total Memory Install	

Software Configuration

Document your software environment as described in the following sections:

- Operating System 3-6
- Applications 3-7

Operating System

List the following operating system information:

- Operating system name
- Release level
- Patch history
- Kernel configuration parameters
- Swap size
- OS-specific software installed
- High availability software installed

You may need to contact your operating system vendor to get system upgrades, patches, or help with problems, so make a note of the following operating system technical support information:

- Telephone number
- Support hours
- Name of your technical account manager and his or her telephone or pager number

- Web page if applicable

Applications

Make a list of applications to be migrated to Adaptive Server 11.5. For each application, note:

- Information about data and usage
  - Distributed data
  - Warehoused or used in transaction processing? If used in transaction processing, it needs to be accurate and in the right format. Warehoused data is more tolerant of format and slight computational differences, such as datatype conversion differences.
- Location of application source files
- Types and number of modules containing SQL code to be evaluated for needed modifications (triggers, stored procedures)

Sybase Configuration

Document your Sybase configuration as described in the following sections:

General Information

Record the following Sybase information:

- SQL Servers and their SYBASE home directories
- Components and release levels (including EBFs)
- names and location of scripts to rebuild database environment
- All SQL Server configuration values
  - Save the output of **sp_configure.** This shows the current settings of all configuration parameters.
  - Run buildmaster -d -yall (System 10 and below).

- SQL Server Runtime Memory Map
  - Run **dbcc memusage** during non-peak time or in single user mode. This command tells you how memory is configured within SQL Server. See the *Performance and Tuning Guide* for more information about ways to capture server memory statistics. You can record the output of **dbcc memusage** in a table like this one:

Table 3-8: *dbcc memusage* output

	Mb	2K Blocks	Bytes
Configured Memory	10.0000	5120	10485760
Code size:	2.7075	1387	2839060
Kernel Structures:	1.5924	816	1669755
Server Structures:	1.8529	949	1942928
Page Cache:	3.0496	1562	3197756
Proc Buffers:	0.0270	14	28348
Proc Headers:	0.6191	317	649216
Number of page buffers:		1489	
Number of proc buffers:		372	

Database Devices

Record database device information, as in the following example:

Table 3-9: Database Devices

Database Device Name	Physical Device name	Mirrored Device Name	Virtual Device Number	Size (Mb)
TRD_dev1	/dev/rdsk/c0t0d0s3		2	10020
TRD_dev2	/dev/rdsk/c0t1d0s3		3	5020
TRD_log	/dev/rdsk/c0t1d0s4		4	1020
CIS_dev1	/dev/rdsk/c0t1d1s3		5	4020
CIS_log	/dev/rdsk/c0t1d1s4		6	420

Databases and Segments

List of all segments and the objects on them. Use a worksheet like that in the following example:

Table 3-10: Objects and Segments

Database Name	DB Options Set	Size (Mb)	Segments Names	Device Name	Size (Mb)
master	none	700	default.system.log	master	3
master	none	500	default.system.log	master	2
master	none	300	default.system.log	master	1
model	none	200	default.system.log	master	2
tempdb	select into/bulkcopy	200	default.system.log	master	2
TRD	none	10000	system, trd_seg1	TRD_dev1	10020
TRD	none	5000	system, trd_seg1, trd_seg2	TRD_dev2	5020
TRD	none	1000	log	TRD_log	1020
CIS	none	4000	system, cis_seg1	CIS_dev1	4020
CIS	none	400	log	CIS_log	420

Dump Devices

Record dump device information as in the following example:

Table 3-11: Dump Devices

Database Device Name	Physical Device name	Media type	Capacity (Mb)
Tape dev1	/dev/rmt/0m	4mm	2000
Tape dev2	/dev/rmt/1m	4mm	2000
Tape dev3	/dev/rmt/2m	4mm	2000
Tape dev4	/dev/rmt/3m	4mm	2000
Tape dev5	/dev/rmt/4m	4mm	2000

SQL Server Objects

Document the objects in your current SQL Server as described in the following sections:

Gather Scripts to Create Objects

Locate or create the scripts necessary to recreate:

- Server level objects
  - Database devices
  - Configurations
  - Logins and security
- Database level objects, including:
  - Defaults, rules, and user datatypes
  - User databases
  - Users, groups, and aliases
  - Tables, views, and stored procedures

- Other database objects such as triggers and indexes

You may also want need to extract and load data with **bcp**. These scripts can be used to help set up your test environment as well as the building a new production system and may be needed if you plan to maintain two server systems at different release levels.

If You Don't Have Scripts

There are several ways you can reproduce scripts or access the information needed to reproduce configuration and objects.

Query Sybase System Tables

The following system tables contain object information that you can use to create installation scripts:

- *sysdatabases*
- *sysdevices*
- sysusages
- *sysobjects*
- *sysusers*
- *sysservers*
- *syslogins*
- *sysremotelogins*

See the *System Administration Guide* for more details on system tables and objects. See also "Segment Remapping with **load database** When Moving a Database" in Technical Information Library for information on using system tables to reconstruct databases.

Use System Stored Procedures

For information about current SQL Server configuration, use **sp_configure** with no argument. It will list all configuration parameters and their values.

For the SQL commands in a stored procedure, use **sp_helptext.**

See the *Reference Manual* for other system stored procedures such as **sp_helpdevice**, **sp_help**, **sp_helpdb, sp_helpsegment** and **sp_helpindex** that can provide information about objects on your server.

Use Sybase Tools

You can reverse engineer server objects using Sybase DBA tools such as SQL Server Manager, Sybase Central, or Powersoft PowerDesigner 6.0. The the manuals for these tools.

Use Third Party Tools

Some third party vendors offer tools that you can use to analyze and write scripts for your objects. For example, Cyrano's Migration Pack can help identify potential technical problems as well as being able to extract object definitions from your databases. Sybase uses Cyrano as part of the SAFE/EM migration methodology. See Cyrano's Web page at http://www.cyrano.com, as well as Appendix A, "The Cyrano Migration Pack" for more information.

4 Prepare: Writing a Plan and Getting Ready to Migrate

In this Chapter

Now that you've collected data about your current system, you can make a plan for migrating. In addition to choosing a migration method, you may need to bring your system resources to the level required by Adaptive Server 11.5 and make changes needed in applications and system administration procedures.

This chapter covers the following issues:

As part of migration planning, see the Sybase Migration Resource Guide at http://www.sybase.com/migration for current offerings.

Advance Reading

Before planning and carrying out the migration, become familiar with Adaptive Server 11.5 migration issues. Good starting points include:

- Information available online at the Sybase Migration Resource Guide at http://www.sybase.com/migration, including information about migration tools, services, and programs.
- The Sybooks guide, *What's New in Sybase Adaptive Server Enterprise 11.5?*
- *Release Bulletin* (for your platform)
- The Adaptive Server installation guide for your platform.

For other documentation and resources, see Chapter 1, "Introduction and Guide to Resources."

Determine Migration Approach

The migration strategy you use will depend on such factors as the cost of the effort, the type of business you do, the size of your databases, available resources.

The Table 4-1 highlights the advantages and disadvantages of each migration approach.

Table 4-1: Migration approaches

Approach	Advantages	Disadvantages	When Used
Parallel With Replication	Easy fallback to release 10.0.2.5 or later. You do not need to rebuild release 10.0.2.5 or later databases. Minimal system down time.	Can be complex in OLTP environments. Replication Server must be set up, requiring extra hardware and software. **You must be at release 10.0.2.5 or later**. You can replicate from release 4.x, but you cannot replicate to release 4.x.	This approach is best for large 7x24 production databases, maintaining high availability, when: • Rebuilding a release 10.0.2.5 or later database can take too long. • The system may have a large number of transactions and complex Transact-SQL queries with subqueries.
Cutover Without Replication	Can be executed with minimal resource demands.	Highest risk. Requires down time for critical migration tasks. Recovery can be time consuming in a production environment.	This approach is likely for resource-constrained environments.
Phased Cutover	Low risk with low development overhead. Especially conducive to testing.	May require additional resources—either more memory or a second system, if both pre-release 11.x and release 11.x servers do not perform acceptably on the current system. Requires tighter coordination with application groups and database owners.	If neither of the other two approaches seems appropriate, you can use a phased cutover.

For more information on these approaches, see the following sections:

- Parallel With Replication 4-3
- Cutover Without Replication 4-5
- Phased Cutover 4-7

➤ ***Note***

This migration guide does not cover other parallel migration approaches; such as, running two systems in parallel where you have to maintain both the systems simultaneously, or transaction duplication where you use one front-end to drive two parallel back-ends. These system operational approaches include factors too site-specific to detail effectively in this guide.

Whenever possible, upgrade **test** and **development** databases release 11.5 first. Upgrade the production system after testing. See Chapter 7, “Test: Ensuring Stability and Performance” for more information about testing.

Parallel With Replication

The issues for the parallel (with replication) approach are described in *Table 4-2: Parallel migration*:

Table 4-2: Parallel migration

Issue	Recommendations and Tips
Method	Install a new copy of Adaptive Server 11.5 Copy in or load pre-release 11.5 databases t Use Replication Server to maintain both sets of databases. The release 11.x system becomes the primary server, and you maintain the pre-release 11.x system as a warm standby.

Table 4-2: Parallel migration (continued)

Issue	Recommendations and Tips
Fallback	Plan for all users to reconnect to SQL Server 10.0.2.5 or later after you take SQL Server 11.5 offline. Make TCP/IP address and port changes where appropriate. Test the fallback process as part of the application test suite. This suite should do both of the following: • Insert data into SQL Server 11.5. The data must be replicated and available in SQL Server 10.0.2.5 or later. • Execute the fallback script. Consider making daily **bcp** dumps of the databases. To fall back, you can then load the dumps into release 10.x. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.0 cannot read release 11.5 backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**. For information about scheduling backups of user databases, see the *System Administration Guide*. Here are some additional tips: • Be sure to upgrade Replication Server first. • Be sure the applications are using the correct server. For details on the interfaces file and the $DSQUERY environment variable, see the configuration guide for your platform. • Remember to include any client fallback in the calculation. • For 7x24 sites, load time delays can impact synchronization. Consider replicating to SQL Server 11.5 and then switching servers.
Application test suite	Since the parallel with replication approach is best for high availability applications, it is imperative that the test suite addresses both update correctness and performance acceptability. Execute this suite *before* switching users to the new system. See Chapter 7, “Test: Ensuring Stability and Performance” for more information on testing. **Note:** After successful validation, consider having users enter production queries with the production toolset. A good time to do so is after hours or during production lulls.
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues. To ensure correct updates and acceptable performance, test the replicated environment. See the *Replication Server Administration Guide.*

Table 4-2: Parallel migration (continued)

Issue	Recommendations and Tips
Environment	The environment used for SQL Server 11.5 needs to be more powerful to handle the query and replication loads. See the *Replication Server Configuration Guide*. Be sure to account for any increased release 11.5 memory requirements that apply to your configuration. For more information, see: • The installation guide for your platform. It gives basic RAM requirements. • Chapter 6, "Implement: Making Required Database Administration Changes". • Details on configuring memory and data caches in the *System Administration Guide* • Information on how to configure memory for performance in the *Performance and Tuning Guide*. **Note:** For a production system, execute the performance suite during off hours.
Scheduling	Developing and running the replication facility, validation and performance suite, and fallback script requires significant effort. If your environment already uses replication, this effort will be notably easier. For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Cutover Without Replication

The issues for the cutover without replication approach are:

Table 4-3: Cutover migration

Issue	Recommendations and Tips
Method	Upgrade all databases to release 11.5 at the same time. A cutover without replication is common in small organizations for development or production servers.

Table 4-3: Cutover migration (continued)

Issue	Recommendations and Tips
Fallback	Base fallback on the amount of time needed to restore earlier databases. For example, if users need the system Monday at 8 a.m. and the restoration takes 8 hours, the validation test must pass by Sunday at midnight. **Note:** Remember to include any client fallback in the calculation. You can use **dump database** or **bcp out** before an upgrade to prepare for fallback. Plan a way to capture transactions after cutover to be used in case of fallback. If you go into production and then need to back off, you will have to restore all the transactions that occurred after the last dump / load.
Application test suite	For a development system, simple validation may be adequate. However, for a **production** system, the test suite must address both update correctness and performance acceptability. See Chapter 7, "Test: Ensuring Stability and Performance" for more information on testing. **Note:** You might want to validate over a 3-day weekend if possible.
Fallback after cutover	Consider making daily **bcp** dumps of the databases. You can then load the dumps into release 10.x to fall back. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.0 servers cannot read release 11.5 backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**. For information about scheduling backups of user databases, see the *System Administration Guide*.
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues.
Environment	Be sure to account for any increased release 11.5 memory requirements that apply to your configuration. For more information, see: • The installation guide for your platform. It gives basic RAM requirements. • Chapter 6, "Implement: Making Required Database Administration Changes". • Details on configuring memory and data caches in the *System Administration Guide* • Information on how to configure memory for performance in the *Performance and Tuning Guide*. **Note:** For a production system, execute the performance suite during off hours.
Scheduling	For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Phased Cutover

The issues for a phased cutover are:

Table 4-4: Phased cutover migration

Issue	Recommendations and Tips
Method	Change only one application and database to release 11.x at a time.
Fallback	Consider making daily **bcp** dumps of the databases. To fall back, you can then load the dumps into release 10.x or 4.x. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**, declarative RI (referential integrity) and IDENTITY columns, until the conversion succeeds. For information about scheduling backups of user databases, see the *System Administration Guide*.
Application test suite	Ensure that the application test suite addresses both update correctness and performance acceptability. Also ensure that you do the following: • Maintain the directories / libraries for both releases. • Make sure the applications are using the correct server. • After successful validation, consider having users enter production queries with the production toolset. A good time to do so is after hours or during production lulls. See Chapter 7, “Test: Ensuring Stability and Performance” for more information on testing.
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues. Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. **Note:** Do not use release 11.x schema enhancements, such as **max_rows_per_page**, declarative RI and IDENTITY columns, until the conversion succeeds.

Table 4-4: Phased cutover migration (continued)

Issue	Recommendations and Tips
Environment	Be sure to account for any increased release 11.x memory requirements that apply to your configuration. For more information, see: • The installation guide for your platform. It gives basic RAM requirements. • Chapter 6, "Implement: Making Required Database Administration Changes". • Details on configuring memory and data caches in the *System Administration Guide* • Information on how to configure memory for performance in the *Performance and Tuning Guide*. Here are some additional tips: • Execute performance measurements on a system with similar capabilities. • For a production system, execute the performance suite during off hours.
Scheduling	For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Write a Migration Plan

Produce a project plan which documents:

- Migration strategy—Which method is most appropriate for your site.
- Fallback—What to do in case the migration fails. The plan you evolve will be site-specific, but some general issues are discussed in Chapter 7, "Test: Ensuring Stability and Performance".
- Application test suite—What validation and performance testing to perform for acceptance. See Chapter 7, "Test: Ensuring Stability and Performance" for guidance.
- Bridging—Ways to minimize the impact to users during the migration. Refer to the business requirements you gathered in Chapter 3, "Analyze: Documenting Your Environment".
- Environment—Additional resources and changes to the environment needed based on the information you gathered in Chapter 3, "Analyze: Documenting Your Environment".
- Scheduling—How much time the migration will take based on the level of complexity and business needs. Refer to the business

requirements you gathered in Chapter 3, “Analyze: Documenting Your Environment”.

Producing the following documentation may also be useful:

- A work breakdown that lists tasks chronologically and assigns them to specific roles like the one in Appendix D, “Sample Migration Task Lists”.
- Specification for application changes. The details of needed application changes are discussed in Chapter 5, “Implement: Making Required Application Changes”.

Build the Adaptive Server Environment

After decided the best migration approach for your system, begin creating or updating the environment for Adaptive Server 11.5. Go to the following sections and chapters for information about preparing your environment:

- Update your applications and database administration procedures. See Chapter 5, “Implement: Making Required Application Changes” and Chapter 6, “Implement: Making Required Database Administration Changes” for details.
- Create a test environment. See Chapter 7, “Test: Ensuring Stability and Performance” for more information on the test environment

Update Hardware Resources

Evaluate hardware resource needs according to your migration approach, such as parallel with replication. You may need a separate system, Replication Server for the high availability environment, a disk farm for backout strategies, and more memory.

The *Release Bulletin* for your platform contains information about any hardware requirements.

See the installation guide for your platform for basic RAM requirements. See Chapter 6, "Implement: Making Required Database Administration Changes" for more information on cache requirements.

See the *System Administration Guide* for information on backups.

Verify Operating System Version and Fix Level

Ensure that the operating system is at the proper version and level to run Adaptive Server 11.5. For migration across machines, verify that the release 11.0.x, 10.x, 4.x server will run under the same operating system level.

For release 11.x operating system requirements and required EBFs (bug fixes), see the *Release Bulletin* for your platform.

➤ ***Note***

If you need to perform an operating system upgrade, do so before migrating. Test the new system to be sure it's working properly to avoid introducing unrelated errors into the migration process.

Create Migration Scripts

Using the scripts you located, wrote, or reverse engineered with third party tools in Chapter 3, "Analyze: Documenting Your Environment", write or edit the scripts that will create your 11.5 Adaptive Server installation. You need:

- Server Level Migration scripts to create the new Adaptive Server environment, including database devices, configuration, and logins and security
- Database Level Migration scripts to create Adaptive Server user databases and database objects such as tables, views, indexes, triggers, groups, users, permissions.

Review Adaptive Server Interoperability with Other Sybase Products

To ensure that the versions of other Sybase products in use at your site are compatible with Adaptive Server 11.5, see Technical Information Library for recent interoperability and compatibility information.

Update Applications and System Administration Procedures

The following chapters, Chapter 5, "Implement: Making Required Application Changes" and Chapter 6, "Implement: Making Required Database Administration Changes" detail the 11.5 issues Migrating to Adaptive Server 11.5 requires you to review your current applications and system administration procedures for any changes that could cause system problems or unexpected processing results.

This is the most time-consuming part of migration preparation and you may choose a method to do this that suits your needs and resources. Possible choices include:

- Writing your own scripts to examine and change applications
- Using third party products such as Cyrano's Migration Kit
- On a development system, upgrading to 11.5 and using some of the tools available such as the stored procedures **sp_checkreswords** and **sp_procqmode**. See the *Reference Manual* for information about these stored procedures.

5 Implement: Making Required Application Changes

This chapter discusses changes to Transact-SQL syntax, query processing, datatypes, objects, and reserved words that can cause application failure or unexpected results. It does not attempt to cover all changes that could affect applications. For a comprehensive listing of changes and new features, see *What's New in Adaptive Server 11.5*.

➤ ***Note***

Changing applications and system administration is the most time-consuming part of migration preparation and this guide does not tell you how to make these changes. You must choose a method to do this that suits your needs and resources. Possible choices include:
—Writing your own scripts to examine and change applications
—Using third party products such as Cyrano's Migration Kit
—On a development system, upgrading to 11.5 and using some of the tools available such as the stored procedures **sp_checkreswords** and **sp_procqmode**. See the *Reference Manual* for information about these stored procedures.

In this Chapter

Go to the section that references your current SQL Server version and read from there to the end of the chapter.

If Your Current Version is 4.x

This section summarizes changes introduced with version 10.0 that affect only those systems currently at version 4.x. If you are at version 4.x, read this section and all following sections. Skip this section if you are upgrading from version 10.x or higher.

This section covers the following issues:

New Reserved Words in Version 10.0

These words are reserved as of release 10.0 and can no longer be used as database object names, nor as user, group, or role names:

Table 5-1: New reserved words in 10.0

arith_overflow	escape	of	rows
at	fetch	only	schema
authorization	foreign	open	shared
cascade	identity	option	some
check	identity_insert	precision	stripe
close	isolation	primary	syb_identity
constraint	key	privileges	syb_restree
current	level	primary	user
cursor	mirror	read	user_option
deallocate	national	references	varying
double	noholdlock	replace	work
endtran	numeric_truncation	role	

See “New Reserved Words in Version 11.0” on page 5-16 and “New Reserved Words in Version 11.5” on page 5-19 for reserved words introduced with subsequent releases.

See “Changing Reserved Words in Your Applications” on page 5-20 and “Using the set Command to Keep Object Names” on page 5-20 for information on how to find and resolve reserved word conflicts.

New Login and Password Protocols

Login / password changes include the following:

- New logins and changed passwords require a minimum 6-byte password. Existing passwords smaller than 6 bytes are left alone during upgrade, but the 6-byte minimum is enforced if you add or change passwords.
- Null passwords are not allowed. You need not change existing passwords. The next invocation of **sp_password** will enforce the new rules.
- Expired passwords can cause problems for applications with embedded passwords. If the **systemwide password expiration** is set to 0 (default), passwords do not expire.

➤ ***Note***

Adaptive Server 11.5 enables you to protect sensitive data with several security features, including user identification and authentication, discretionary access controls, division of roles, and event auditing. See the *Security Administration Guide* and the *Security Features User's Guide*.

Datatype Changes

The following sections discuss changes to and additions to datatypes:

New Datatypes and New Default

Two new datatypes, *numeric* and *decimal*, were introduced in version 10.0. Unlike *float*, they are platform independent and exact. An exact numeric result accommodates user-defined ranges and storage sizes. There is less risk from loss of scale due to platform differences. However, applications may break because functions return different datatypes than before.

Adaptive Server now defaults to the *numeric* datatype rather than *float*. For example, this constant:

```
5.1
```

is treated as numeric. If you want to use *float*, represent the constant as

```
5.1e0
```

In addition, the optimizer has trouble with clauses where *float* columns are joined to *numeric* arguments. Table 5-2 shows some examples of different representations of constants as a result of this change:

Table 5-2: Different results with numeric rather than float datatype

Statement	4.9.2	10.0 and later
select 2147483648	integer overflow	2147483648
select 2 * 1.2345678	2.246914	2.24691356
select 2.8 * 5	14.000000	14.0

➤ ***Note***

Pre-release 10.x Client-Library applications running against 10.0 or higher SQL Servers map *numeric* to *float*.

The mathematical functions that return a value of type *numeric* rather than *float* are:

- **abs**
- **ceiling**
- **degrees**
- **floor**
- **power**
- **radians**
- **round**
- **sign**

Datatype Hierarchy

The datatype hierarchy determines the datatype of computational results. The result value is assigned the datatype of its parent component closest to the top of the hierarchy. There is no loss of precision due to conversions. However, Adaptive Server may return datatypes different than those returned by earlier versions of SQL Server.

In release 4.x, the *money* datatype was higher than *float* datatypes. For ANSI compatibility, release 10.x moves *money* below the *float* and *numeric* datatypes. This will cause a *float* value to be returned when both these datatypes appear in a query.

Table 5-3 shows the current datatype hierarchy:

Table 5-3: Datatype hierarchy

Name	Hierarchy
floatn	1
float	2
datetimn	3
datetime	4
real	5
numericn	6
numeric	7
decimaln	8
decimal	9
moneyn	10
money	11
smallmoney	12
smalldatetime	13
intn	14
int	15
smallint	16
tinyint	17
bit	18
varchar	19

Table 5-3: Datatype hierarchy

Name	Hierarchy
sysname	19
nvarchar	19
char	20
nchar	20
varbinary	21
timestamp	21
binary	22
text	23
image	24

Example

In release 4.x, *money* was above *float* in the hierarchy. It is now below both *float* and *numeric*. For example, the following query:

```
select $12 * 8.9
```

returns a result of type *numeric*. In release 4.x, this query returned *money*. Likewise, the following query:

```
select $12 * 8.9e0
```

returns a result of type *float*. In release 4.x, this query returned *money*.

If you want the release 4.x behavior you must use **convert**:

```
select $12 * convert(money,$12 * 8.9e0)
```

Datatype Conversions

The following changes apply to datatype conversions:

- **Numeric to character conversion**: release 4.x *float*-to-*character* conversion allowed some truncation without warning. In release 10.x and 11.x, all conversions to character succeed only if no decimal digits are lost. Previously, floating point to character conversions allowed some truncation without warning.
- **Conversion to money**: All conversions to *money* datatypes round to four places.
- **Explicit conversion from numeric to numeric**: When an explicit conversion of one numeric value to another results in loss of

scale, the results are truncated without warning. For example, explicitly converting a *float* to an *integer* causes SQL Server to truncate all values to the right of the decimal point.

The rationale is that explicit conversions are done purposefully with an understanding of the implications and consequences.

- **Integer to character conversion**: Conversions from integer to character return an error if an overflow occurs. They formerly returned a buffer of "***".

ANSI SQL92 Support for Conversion Error Handling

Conversion error handling is now ANSI SQL92 compliant. These changes allow an application to decide the severity of error. They may result in different error handling behavior than in release 4.x. Here is a summary of these changes:

arithabort arith_overflow

The command:

```
set arithabort arith_overflow
```

specifies behavior following a divide-by-zero error or a loss of precision during either an explicit or an implicit datatype conversion. (In 4.x, divide-by-zero returned NULL.) Options are:

ON (default): Rolls back the entire transaction or batch in which the error occurs.

OFF (ANSI SQL92 Standard): Aborts the statement that causes the error and continues to process other statements in the transaction or batch.

Application recoding is required to use ANSI semantics.

arithabort numeric_truncation

The command:

```
set arithabort numeric_truncation
```

specifies behavior following a loss of scale by an exact numeric type.

Options are:

ON (Default and ANSI SQL92 Standard): Aborts the statement that caused the error continues to process other statements in the transaction or batch.

OFF: truncates value and continues processing.

arithignore arith_overflow

The command:

```
set arithignore arith_overflow
```

specifies whether error messages are sent upon numeric overflow

The command options are:

ON: Doesn't advise application of overflow or divide-by-zero errors

OFF (Default): Advises application of overflow or divide-by-zero errors

Subquery Processing Changes

This section describes changes in subquery processing. The examples used are from the *pubs2* sample database supplied with your software. For more information about *pubs2* and its tables, see the *Reference Supplement*. Changes to query processing include:

in and any

Subqueries with **in** and **any** no longer return duplicate rows. In version 4.x, SQL Server processed the query in the example below as a join and returned one row from the outer table for each matching row from the inner table.

However, the semantics of ANSI existence tests prevent rows from the outer query being returned more than once. If your application requires duplicate rows, it will break.

This example, taken from the *pubs2* sample database, shows how result sets changed between 4.x and 10.0:

```
select pub_name from publishers
where pub_id in
(select pub_id from titles)
```

Release 4.x Results	Release 10.x and 11.x Results
New Age Books	New Age Books
New Age Books	Binnet & Hardley
New Age Books	Algodata Infosystems
New Age Books	
New Age Books	
Binnet & Hardley	
Binnet & Hardley	
Binnet & Hardley	
Binnet & Hardley	
Binnet & Hardley	
Binnet & Hardley	
Binnet & Hardley	
Algodata Infosystems	
Algodata Infosystems	
Algodata Infosystems	
Algodata Infosystems	
Algodata Infosystems	
Algodata Infosystems	

not in

SQL Server 4.x returned “true” if a subquery using **not in** returned results contained no matching values but did contain a NULL. Though NULL usually means “unknown”, ANSI semantics define NULL as “false”. Correlated subqueries using **not in** now return “false” when the subquery returns NULL.

This change allows you to get correct results from subqueries using **not in** without writing extra code.

This example shows the difference in processing due to this change:

```
select pub_id
from publishers
where $100.00 not in
(select price from titles
where titles.pub_id=
publishers.pub_id)
```

Release 4.x Results	Release 10.x and 11.x Results
0736 0877 1389	0736

or...exists/in/any

Subqueries that contain **exists**, **in**, or **any** under an **or clause** now return correct results sets. SQL Server 4.x did not return rows when a subquery evaluated to "false", even if the **or** clause was "true".

You now get correct results without having to write extra code.

This example from *pubs2* shows the difference in results when a query contains **in** and a true **or** clause:

```
select pub_name
from publishers
where pub_id in
(select pub_id from titles
where title = "No Such Book")
or pub_id = '1389'
```

Release 4.x Results	Release 10.x and 11.x Results
NULL	Algodata Infosystems

>all and <all

Subqueries that contain **>all** and **<all** now return the correct results. ANSI semantics evaluate these subqueries as "true" when they match no rows.

This example shows how different releases handle a query that compares advance amounts paid by a non-existent publisher name:

```
select title
from titles t1
where t1.advance > all
(select advance
from publishers p, titles t2
where
p.pub_name="No Such Publisher"
and t2.pub_id = p.pub_id)
```

Release 4.x Results	Release 10.x and 11.x Results
NULL	all rows in the *titles* table

Correlated Subqueries

Release 4.x erroneously suppressed duplicates if all columns in the outer query also were used in the subquery.

If you rewrite release 4.x subqueries for release 11.5, they will return correct results. As a result of the new behavior:

- The client has additional rows to process.
- The application has to be able to handle the duplicates.

This example uses tables from *pubs2* to compare results returned by different releases:

```
select pub_id,
(select count(*) from publishers
where publishers.pub_id = titles.pub_id
```

Release 4.x Results	Release 10.x and 11.x Results
0736 5	0736 5
0877 7	0736 5
1389 6	0736 5
	0736 5
	0736 5
	0877 7
	0877 7
	0877 7
	0877 7
	0877 7
	0877 7
	0877 7
	1389 6
	1389 6
	1389 6
	1389 6
	1389 6
	1389 6

Aggregates with *exists*

SQL Server 4.x sometimes returned the wrong answer when a query contained an aggregate, a correlated subquery contained an **exists** predicate, and the table in the subquery's **from** clause contained duplicates. This happened because the server processed the query as a join and counted each matching row.

As of version 10.0, you get correct results without having to write extra code. There are no risks associated with this change.

This example from *pubs2* compares results for this type of query:

```
select count(*)
from publishers
where exists
(select * from titles
where titles.pub_id = publishers.pub_id)
```

Release 4.x Results	Release 10.x and 11.x Results
18	3

In release 4.x, this query returns 18 items, including 15 duplicates; in release 10.x and 11.x, this query returns 3 items.

select distinct

Prior to release 10.x, correlated **in** subqueries using **distinct** would cause the outer query to return no rows. These subqueries now return the correct results.

This example compares releases:

```
select pub_name
from publishers
where pub_id in
(select distinct pub_id from titles
where titles.pub_id = publishers.pub_id)
```

Release 4.x Results	Release 10.x and 11.x Results
NULL	New Age Books Binnet & Hardley Algodata Infosystems

Additional ANSI-Related Transact-SQL Changes

Other changes to Transact-SQL to bring it into ANSI compliance include:

between

In statements with **between**, ANSI standards require that the values used be given in the normal order. Therefore:

```
expr1 between expr2 and expr3
```

must mean this:

```
expr2 <= expr1 <= expr3
```

In version 4.x, SQL Server switched *expr2* and *expr3* automatically at compile time if it knew that *expr2* was greater than *expr3*. In other words, it would process "250 between 400 and 200" as though it were the same as "250 between 200 and 400"

As of version 10.0, SQL Server processes such inverted statements as "false".

Here is an example that compares releases:

```
create table demo (id int)
insert into demo values (250)
select id from demo
where id between 400 and 200
```

Release 4.x Results	Release 10.x and 11.x Results
250	no rows returned

ANSI Comments

ANSI comments start with two or more consecutive hyphens (--). They terminate in a new line. SQL Server adopted this convention in 10.0. This may cause come current applications to break if they contain syntax such as:

```
select 5--2
```

To avoid this problem, Rewrite these types of expressions with either of the following:

- Parentheses

  ```
  select 5 - (-2)
  ```

- Extra spaces

  ```
  select 5 - -2
  ```

Correlation Names

ANSI requires correlation names on self-joins. The following query is now invalid:

```
select columns from table1, table1
where clause ...
```

This query must be rewritten as:

```
select columns from table1 t1, table1 t2
where clause ...
```

When used, correlation names must be used throughout the query.

For example, the following query is now invalid

```
select title_id from titles t
where titles.type = trad_cook
```

It must be rewritten as:

```
select title_id from titles t
where t.type = trad_cook
```

Mixing correlation names within subqueries, though not an error, may bring different results:

```
select * from my_table where columnA =
select min(columnB) from my_table m where
my_table.columnC = 10)
```

➤ ***Note***

You should use explicit correlation names throughout your syntax. This makes your applications easier to maintain. If you use explicit correlation names in the **from** clause, use the same names in the **select** list and **where** clause.

select into and NULL Column Headings

SQL Server 4.x allowed null column headings in tables created using the **select into** command. This can make SQL code and result sets difficult to read, and is not allowed under ANSI.

Versions 10 and 11 now require column headings to be valid SQL identifiers.

Examples of select list items requiring column headings are:

- Aggregate functions such as **avg(advance)**
- Arithmetic expressions such as *column_name* * 2
- String concatenation such as *au_lname* +, + *au_fname*
- Built-in functions such as **substring(au_lname,1,5)**
- Constants such as results

Here are some syntax examples that specify column headings:

```
select title_id, avg_advance = avg(advance)
into #tempdata from titles

select title_id, avg(advance) avg_advance
into #tempdata from titles

select title_id, avg(advance) as avg_advance
into #tempdata from titles
```

If Your Current Version is 4.x or 10.x

Continue with this section if you are upgrading from SQL Server 4.x. Begin here if you are upgrading from 10.x.

Skip this section if you are upgrading from version 11.0.x.

This section covers the following issues:

New Reserved Words in Version 11.0

The following words are reserved as of version 11.0 and can no longer be used as database object names, nor as user, group, or role names:

Table 5-4: New reserved words in 11.0

max_rows_per_page
syb_terminate
partition
unpartition
online

See “New Reserved Words in Version 11.5” on page 5-19 for reserved words introduced in the next release.

See “Changing Reserved Words in Your Applications” on page 5-20 and “Using the set Command to Keep Object Names” on page 5-20 for information on how to find and resolve reserved word conflicts.

Subquery Processing Changes

Changes in subquery processing are described in the following sections:

Expression Subqueries

Expression subqueries may be slower in release 11.x than 10.x where:

- The outer table is very large and has few duplicate correlation values.
- The inner table is small.
- The subquery contains an aggregate.

The optimizer will not flatten this type of query to be processed as a join. Such a query might look like this:

```
select * from huge_table where col_x=
    (select sum(col_a) from tiny_table
    where col_b = huge_table.col_y)
```

To get faster results, you can reformulate the query to mimic the behavior of release 10.x, as demonstrated in this example:

```
select huge_table.col_y, s=sum(col_a)
    into #t
    from huge_table, tiny_table
    where col_b=huge_table.col_y
    group by huge_table.col_y

select huge_table.*
    from huge_table, #t
    where col_x=#t.s
    and huge_table.col_y=#t.col_y
```

No *set dup in subquery*

The change in the **set dup in subquery** command does not affect systems being upgraded from 4.x, as this command was introduced in

version 10.0 and is now obsolete. Applications that use it receive a warning message and subqueries no longer return duplicates.

You may have used this option to obtain better performance. Because of subquery processing changes, you must rewrite your query as a join if you want duplicates. You should see better performance in Adaptive Server for these types of queries.

union Limitations

Only 16 subqueries are allowed on one side of a **union**. This does not affect most queries because only 16 tables are allowed within one query. This only affects a query with more than 16 subqueries where some of those subqueries have no **from** clauses.

Subqueries and NULL Results

Prior to SQL Server 11.x, a correlated expression subquery in the **set** clause of an **update** returned 0 instead of NULL when there were no matching rows. SQL Server 11.x correctly returns NULL when there are no matching rows, and raises an error.

For example, the following trigger tries to update a column that does not permit NULL values:

```
update t1
    set c1 = (select max(c1)
    from inserted where t1.c2 = inserted.c2)
```

The correct trigger is:

```
update t1
    set c1 = (select isnull(max(c1), 0)
    from inserted
    where t1.c2 = inserted.c2)
```

The **where** clause updates *t1.c1* to 0, if the subquery does not return any correlation values from the outer table *t1*.

No Subqueries in Updatable Cursors

The following constructs are no longer allowed in the **select** statement of updatable cursors:

- Subquery
- **distinct** option
- **group by** clause

- Aggregate functions
- union operators
- **at isolation read uncommitted**

See the *Transact-SQL User's Guide* for details about using updatable cursors.

If Your Current Version is 4.x, 10.x, or 11.0.x

This section describes changes introduced with version 11.5 that affect all systems at lower versions. Read this section and all following sections.

If you are at release 11.0.x, you do not have to worry about application failure due to syntax changes in Transact-SQL. This section covers the following issues:

➤ ***Note***

Consider application modifications to take advantage of Adaptive Server 11.5 performance-enhancing features. *See What's New in Adaptive Server Enterprise 11.5* for a comprehensive overview of changes and new features.

New Reserved Words in Version 11.5

This section covers the following issues:

Reserved Words in 11.5

The following words are reserved as of version 11.5 and can no longer be used as database object, user, group, or role names:

Table 5-5: New reserved words in 11.5

activation	identity_start
connect	membership
consumers	passwd
exclusive	proxy
external	session

Changing Reserved Words in Your Applications

Sybase-provided tools such as the reserved words check option of the upgrade program for UNIX and the stored procedure **sp_checkreswords** allow you to check for database name conflicts in Adaptive Server both before and after upgrade. However, you must change the references to these objects in your applications as well to prevent processing failures.

Checking for reserved words in your applications may be a laborious process. You can write your own scripts to perform this check. An easier option may be using third-party tools such as the Cyrano Migration Pack which contain utilities to help you identify potential problems before upgrading. See Appendix A, "The Cyrano Migration Pack" for information about the migration tools created by Cyrano for Sybase migration.

Avoiding Future Reserved Word Conflicts

To avoid having to search for reserved words in new releases, you can adopt a naming convention where you use an initial and underbar with every object name. For example, a table named *user* can be written *u_user*, and a column in the table named *address* can be written *u_address*.

Using the *set* Command to Keep Object Names

If you choose not to change object names, you can use the **set quoted_identifier** option. You must add the following **set** command to all your applications, putting quotes around all keywords, when you issue Transact-SQL statements. For example:

```
set quoted_identifier on
select "user" from table_x
```

For more information, see the section on the **set** command in the *Adaptive Server Enterprise Reference Manual*.

Sorted Order in Queries with Clustered Indexes

Queries on tables with clustered indexes have, in past releases, returned results in a predictable order, that is, index order. However, this behavior was never guaranteed by **select** semantics. and was only an artifact of linear processing.

Adaptive Server may now perform a parallel scan on a clustered index if this is the best plan. In this case, results may not return in the same order as before; they may, in fact, differ in order between different executions of the same query. If the query specifies **rowcount**, the results may be completely different. This is not an error.

If you need results returned in a certain order, do not rely on the existence of a clustered index. Use an explicit **order by** clause.

Changes Due to Two-Phase Commit Enhancements

If you upgraded your existing two-phase commit configuration to the new 11.5 configuration, make sure that user applications or stored procedures reference *sybsystemdb..spt_committab* instead of *master..spt_committab*. For more information about changes to two-phase commit, see Chapter 6, “Implement: Making Required Database Administration Changes”, as well as *What’s New in Sybase Adaptive Server Enterprise 11.5*.

A Few Transact-SQL Programming Tips

How you code can affect optimizer decisions. The following sections contain a few programming tips that may improve your processing speed:

Syntax That Can Slow You Down

The following examples show syntax that can take longer to process than some simple alternatives:

- Avoid mathematical manipulation of indexed columns when using search arguments. For example, do not use mathematical functions on the column itself, as shown in the example. Instead, perform the function against the constant on the other side of the operator.

  Don't:

  ```
  select name from employee
  where salary * 12 > 100000
  ```

  Do:

  ```
  select name
  from employee
  where monthly_salary > 100000 / 12
  ```

  Avoid use of any function against the indexed column(s) on the "left" side of the operator.

- Avoid incompatible datatypes in column names, search arguments, or stored procedure parameters. Keep datatypes consistent throughout an application or server. Incompatible datatypes include:

  - *float* and *integer*
  - *char* NOT NULL and *varchar*
  - *binary* and *varbinary*

  However, *integer* and *intn* (allows NULLs) are compatible

- Multiple **or** clauses. **or** clauses are expensive for these reasons:

  - If any portion of the **or** clause requires a table scan, the whole query will use table scanning.
  - **or** clauses require creating and sorting a work table

  Consider using **unions** as an alternative.

- Use **group by** instead of using **distinct** without aggregates. This reduces the size of the worktable created and, by storing distinct values only in the **group by** worktable, reduces the processing resources needed to sort them.

  Don't:

  ```
  select DISTINCT ship_to_address
  from customer_orders
  where last_order_date >= "01/01/1997"
  ```

  Do:

```
select ship_to_address
from customer_orders
where last_order_date >= "01/01/1997"
GROUP BY ship_to_address
```

- Use the leading column of an index. If the leading column is missing from the query, Adaptive Server can't utilize the index.

  In this example, Table A has a non-clustered index on columns ColA, ColB, and ColC.

  Coded this way, the query will not use the index:

```
select ColA, ColB, ColC, ColD, ColE
from tableA
where ColB = 45
and ColC = "Quality"

select ColA, ColB, ColC
from tableA
where ColC = 45
```

  Coded this way, the query may be able to use the index because the index may "cover" the query (the index may supply all the data required):

```
select ColA, ColC
from tableA
where ColB = 45
```

  Coded this way, the query will use the index to find the specific data, since each **select** clause uses one or more leading columns:

```
select ColA, ColC, ColD, ColE
from tableA
where ColA = 45

select ColA, ColC
from tableA
where ColA = "C"
and ColB = 99

select ColA, ColC
from tableA
where ColA = "A"
and ColC = "Quality"

select ColA, ColC, ColD
from tableA
where ColA = "A"
and ColB = 401
and ColC = "Quantity"
```

General Tips

- Try to use queries that can be "flattened" by the optimizer. The optimizer can treat certain subqueries like normal or existence joins, which are much quicker to execute. Queries that can be flattened include:
  - Many **in**, **any**, and **exists** subqueries
  - Expression subqueries like the following:

    ```
    column {<, <=, >, >=, !=, =} subquery
    ```

  - Expression subqueries with unique joins or that return unique columns.
- Avoid subqueries that cannot be flattened, such as the following:
  - Most **not in**, **not exists**, and **all** subqueries
  - **in**, **any**, **exists** in an **or** clause
  - **in**, **any**, **exists** in a correlated subquery with aggregates
  - Expression subqueries without unique joins or not returning unique columns
- Queries with **exists** and **not exists** are faster than **in** and **not in** when used with **if**. The **exists** test will stop processing the instant it finds a matching row. This is true for both **if** statements and subqueries.

  Don't:

  ```
  if 0 <  (select count(*) from Employee where
  LastName = "BURKE")
  begin
    .../*statement group*/
  end
  if "BURKE" not in (select LastName from Employee)
  begin
    .../*statement group*/
  end
  ```

  Do:

  ```
  if exists (select * from Employee where LastName
  = "BURKE")
  begin
    .../*statement group*/
  end
  ```

- Use ">=" rather than ">" when using a non-unique index or just the leading columns of a composite index. In a query such as this (with a non-unique index on the age column of Employee. In a

query such as the example, the query process must scan all index pages where age = 30 to find the first qualifying row that is larger:

Don't:

```
select EmployeeName
from Employee
where age > 30
```

However, when the same query is written in either of two ways, the query process goes directly to the first qualified row and starts processing from there:

Do:

```
select EmployeeName
from Employee
where age >= 30 + 1
```

Or do:

```
select EmployeeName
from Employee
where age >= 31
```

Learn More About Coding For Performance

To learn more about programming for performance, as well as monitoring and tuning your system, consider taking Sybase's "Migrating to Adaptive Server 11.5" course. For more information, see Chapter 1, "Introduction and Guide to Resources".

6 Implement: Making Required Database Administration Changes

In this Chapter

This chapter discusses changes to Adaptive Server system administration that can cause problems if you are not prepared for them. It does not attempt to cover all new, helpful features. For a comprehensive listing of changes and new features, see *What's New in Adaptive Server 11.5*.

➤ ***Note***

Changing applications and system administration is the most time-consuming part of migration preparation and this guide does not tell you how to make these changes. You must choose a method to do this that suits your needs and resources. Possible choices include:
—Writing your own scripts to examine and change applications
—Using third party products such as Cyrano's Migration Kit
—On a development system, upgrading to 11.5 and using some of the tools available such as the stored procedures **sp_checkreswords** and **sp_procmode**. See the *Reference Manual* for information about these stored procedures.

The information in the sections below is cumulative. Start at the section that references your current version and work through to the end of the chapter.

If Your Current Version is 4.x

If your current version is SQL Server 4.x, begin with this section and continue to the end of the chapter.

This section covers the following issues:

New Reserved Words in Version 10.0

These words are reserved as of release 10.0.x and can no longer be used as database object, user, group, or role names:

Table 6-1: New reserved words from 4.x to 10.0

arith_overflow	escape	of	rows
at	fetch	only	schema
authorization	foreign	open	shared
cascade	identity	option	some
check	identity_insert	precision	stripe
close	isolation	primary	syb_identity
constraint	key	privileges	syb_restree
current	level	primary	user
cursor	mirror	read	user_option
deallocate	national	references	varying
double	noholdlock	replace	work
endtran	numeric_truncation	role	

Login and Password Changes

Login and password changes include the following:

- New logins and password changes require a minimum 6-byte password. Existing passwords smaller than 6 bytes are allowed during upgrade, but the 6-byte minimum is enforced if you add or change passwords.
- Null passwords are not allowed.
- Expired passwords can cause problems for applications with embedded passwords. If the **systemwide password expiration** is set to 0, the default, passwords do not expire.

- You need not change existing passwords; however Adaptive Server begins enforcing the new password rules the first time **sp_password** is invoked.

➤ ***Note***

SQL Server 11.1 enables you to protect sensitive data with several security features, including user identification and authentication, discretionary access controls, division of roles, and event auditing. See the *Security Administration Guide* and the *Security Features User's Guide*.

The following changes to password security should be noted:

- NULL passwords are no long allowed
- All passwords must be a minimum of 6 bytes
- Passwords expire after a configurable length of time.

RUN File Renamed

On UNIX platforms, **startserver** looks for the default file called *RUN_SYBASE,* rather than *RUNSERVER*. If a file called *RUNSERVER* resides in the *install* directory, **sybinit** changes its name to *RUN_SYBASE* during upgrade.

Changes you may need to make are as follows:

- If you use the **startserver -f RUNSERVER** command to start your server, you must change it to **startserver -f RUN_SYBASE.**
- If you start the server automatically when you boot your machine, change your system start-up file if necessary.

System Stored Procedure Database

As of SQL Server 10.0, a new system database, *sybsystemprocs*, contains all system stored procedures. This change may impact your system in the following ways:

- Non-default permissions on system procedures must be explicitly changed in *sybsystemprocs.*
- Scripts that reference catalog tables must explicitly reference them as *master.dbo.table_name.*
- If user procedures are put here, they must explicitly reference user tables as *database.owner.table* or *database..table.*

- Procedures that affect the master database cannot be run from within a transaction. For example, you cannot run a procedure such as this one from within a transaction:

  ```
  if @@trancount > 0 print
  ```

- You may need to increase the **open databases** parameter. This parameter must be set to at least the number of databases on your system for the upgrade to succeed.

◆ ***WARNING!***

If the open databases parameter is set to a value lower than the number of databases on your system, the upgrade will fail.

- You need to develop a regular backup plan for *sybsystemprocs*.

Creating the System Stored Procedure Database

The Adaptive Server upgrade program automatically initializes a database device and creates the stored procedure database at the location that you indicate. The database size for release 11.5 is 60Mb. See the installation guide to verify the size so that you can reserve sufficient space.

Moving User Stored Procedures

If you have your own stored procedures, you may want to move them to the new *sybsystemprocs* database, although it is not required. If you do decide to move them, follow these steps:

1. Add the database name and owner to any tables you reference that exist in *master*.
2. Explicitly change the nondefault permissions on system procedures.
3. Explicitly reference scripts that reference catalog tables as *master.dbo.<table_name>*.

New *create table* Permission

In release 10.x and 11.x SQL Server, **create table** permission is explicitly granted for all users on *tempdb*. This permission is granted at server start-up time.

New Backup Server

As of 10.0, Backup Server manages all dumps. All backup and recovery procedures are executed through remote procedure calls (RPCs) between Adaptive Server and Backup Server. To allow these procedures to execute, the *allow remote access config* parameter must be set to "on".

The following changes may affect your dump scripts:

- Dumps to the null device (Unix: */dev/null* or VMS devices starting with NL) are now prohibited
- If you have tapes without ANSI labels, existing dump commands will overwrite the contents.
- If you have single file media (for example, QIC) with ANSI labels, the dump command will behave as follows:
  - If the tape has expired, the contents will be overwritten without warning
  - If the tape has not expired, you will be prompted to confirm the dump location or change tapes
- If you have multi-file media, you must append the **with init** clause to the dump command if you wish to overwrite the contents. Otherwise the current dump will be appended to the end of the tape.

Last Chance Threshold

The threshold manager is a new tool, as of release 10.0, for managing space in segments, particularly the log segment. When you upgrade all existing databases where the transaction log is on its own segment, SQL Server installs a last chance threshold on the log segment.

You need to decide how to use the last chance threshold before you migrate your production system. The default behavior suspends action when the last chance threshold on the transaction log is reached, and all users hang until some action is taken.

You must create the **sp_thresholdaction** stored procedure to define an action, such as **dump transaction**, when the last chance threshold is reached. Otherwise, when you run out of space in your log, all users hang indefinitely. Test a last chance threshold procedure before you upgrade your production databases.

To learn how to manage free space with thresholds, see the *System Administration Guide.*

Sample Procedure

The following code fragment is a sample threshold procedure that you can tune to suit your installation. This sample creates a threshold procedure that dumps the transaction log to a tape device when the last chance threshold is reached:

```
create procedure sp_thresholdaction
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
as
dump transaction @dbname to "/dev/rmt4"
/*
** if this is last-chance threshold, print LOG FULL
** @status is 1 (last-chance) or 0 (all others)
*/
if (@status&1) = 1
begin
    print "LOG FULL: database '%1!'", @dbname
end
```

Open Transactions

Even if a last chance threshold procedure exists to dump the transaction log, a problem occurs if an open transaction is filling the log. The **dump transaction** command does not clear the log because of the open transaction.

➤ ***Note***

The new *sysloghold* table enables you to identify open transactions. For more information about this table, see the *Reference Supplement*.

The user who has the open transaction remains in suspend state, keeping the transaction open. If this occurs, you can use the **lct_admin unsuspend** function.

For information about **lct_admin**, see the *SQL Server Reference Manual.* It also describes how to set thresholds with **sp_addthreshold.**

Troubleshooting

When **sp_who** shows the status LOG SUSPEND, you know that users are hung and have reached the last chance threshold. SQL Server writes messages to the error log, listing the tasks that are sleeping because the log is full.

You can change the last chance threshold behavior for a database to the old behavior of "abort the transaction with an 1105 error" by setting "abort xact when log is full" with **sp_dboption.**

Tip for Bulk Copy Users

Bulk Copy handles input records in batches, whether or not the **bcp** inserts are logged. When loading a large number of records, use the **bcp ... in ... -b records** batch option, and set the number of records to a reasonable value, for example, 10,000. This reduces the chance that a **bcp** command will hold a long transaction and block **dump transaction** from clearing enough log space.

If Your Current Version is 4.x or 10.x

If your current version is SQL Server 10.x, begin with this section and continue to the end of the chapter. If your current version is 4.x, continue with this section.

This section covers the following issues:

New Reserved Words in Version 11.0

The following words are reserved as of version 11.0 and can no longer be used as database object, user, group, or role names:

Table 6-2: New reserved words in 11.0

max_rows_per_page
syb_terminate
partition
unpartition

Table 6-2: New reserved words in 11.0

online

Configuration Interface

Prior to release 11.x, SQL Server stored configuration values in the first page of the master device, known as the “configuration block”. Most of these values have been moved to the configuration file, $SYBASE/*SERVER_NAME*.cfg.

Setting Configuration Parameters

Beginning with release 11.0, you can change your server configuration by:

- Issuing an **sp_configure** command
- Editing the configuration file

You no longer need:

- **buildmaster -c**
- **dbcc tune**
- **reconfigure** command

The **reconfigure** command is ignored in this release, but you should remove it from your scripts to prevent problems in future releases.

Configuration File Administration

Backing up the *master* database does not back up the configuration file. You must either:

- Back up the configuration file separately and restore it when you load the *master* database, or
- After loading a *master* database:
  - Start SQL Server.
  - Issue **sp_configure** with the **restore** option.
  - Shut down and restart SQL Server.

New Names for Existing Parameters

Many configuration parameter names changed in version 11.0. If you have scripts that use **sp_configure** to set or report on configuration parameters, you may need to change them if the parameter names have changed.

The table below lists some of the more common parameter names:

Table 6-3: Old and new configuration parameter names

Old Name	New Name
allow updates	allow updates to system tables
cguardsz	stack guard size
cclkrate	sql server clock tick length
cindextrips	number of index trips
cmaxscheds	i/o polling process count
cnblkio	disk i/o structures
cnmaxaio_server	max async i/os per server
cnmaxaio_engine	max async i/os per engine
coamtrips	number of oam trips
cpreallocext	number of pre-allocated extents
cschedspins	runnable process search count
csortbufsize	number of sort buffers
csortpgcount	sort page count
ctimemax	cpu grace time
maximum network packet size	max network packet size
extent i/o buffers	number of extent i/o buffers
fillfactor	default fill factor
user connections	number of user connections
locks	number of locks
memory	total memory
open objects	number of open objects
procedure cache	procedure cache percent
recovery interval	recovery interval in minutes

Table 6-3: Old and new configuration parameter names

Old Name	New Name
remote access	allow remote access
remote connections	number of remote connections
remote logins	number of remote logins
T1204 (trace flag)	print deadlock information
T1603 (trace flag)	allow sql server async i/o
T1610 (trace flag)	tcp no delay

Databases Online / Offline

Beginning with SQL Server 11.0, as part of the automatic upgrade mechanism, a database has two states, *online* and *offline*.

Issuing a **load database** command takes a database offline. If you have scripts that load databases, you must add an **online database** command to your script to make the database available again after the load sequence completes. A load sequence consists of:

- A **load database** execution
- A complete set of **load transaction** statements

isnull Function

Server objects containing the **isnull** function in their code may have to be dropped and recreated. The **isnull** issue affects only upgrades from:

- 10.0.2 through 10.0.2.6
- 11.0 through 11.0.2

For a description of the issue, see “isnull Function in Object Code” on page 6-11 in the section “If Your Current Version is 4.x, 10.x or 11.0.x”.

If Your Current Version is 4.x, 10.x or 11.0.x

If your current version is SQL Server 11.0.x, begin with this section and continue to the end of the chapter. If your current version is 4.x or 10.x, continue with this section.

This section covers the following issues:

New Reserved Words in Version 11.5

The following words are reserved as of version 11.0 and can no longer be used as database object, user, group, or role names:

Table 6-4: New reserved words in 11.5

activation	identity_start
connect	membership
consumers	passwd
exclusive	proxy
external	session

isnull Function in Object Code

The **isnull** issue affects only upgrades from:

- 10.0.2 through 10.0.2.6
- 11.0 through 11.0.2

The **isnull** system function allows you to substitute a value for a NULL in certain instances, for example when inserting a value from a column that contains a NULL to a column that is NOT NULL. The syntax is:

```
isnull(expression1, expression2)
```

where expression1 is potentially NULL and expression2 is the value to substitute. For example, you might use the function to substitute zero for NULL as follows:

```
isnull(interest, 0)
```

When moving from an affected release to 11.5, you may encounter a problem when expression1 and expression2 are of differing datatypes. The issue arises only when the **isnull** function is used in Transact-SQL code that results in a stored query tree, as in stored

procedures, views, triggers, rules, and defaults. The problem is due to a bug that caused query trees of objects containing **isnull** to be stored incorrectly.

To see which objects contain the **isnull** function, use this command:

```
select name, type from sysobjects
where type in ('V', 'P', 'R', 'D', 'TR')
and id in (select distinct id from syscomments
where lower(text) like '%isnull%')
order by type, name
```

Identify those objects that contain **isnull** with differing datatypes. Datatypes to be aware of include tinyint, smallint, numeric, decimal, float(n), real, double precision, smallmoney, or money. After upgrading, drop and recreate those objects.

For more information, see Technical Information Library.

Increased Memory Requirements

This section provides information relevant to all upgrades from earlier releases. Memory requirements have increased significantly between SQL Server 4.x, 10.x, and 11.0.x, and Adaptive Server 11.5.

The increased memory needs reflect the following:

- Growth of the Adaptive Server binary
- Addition of XP Server and Omni services
- New structures in memory, such as user log cache (new in 11.0) and metadata cache
- Growth of old structures such as locks, user connections, and open databases
- Growth of compiled objects in procedure cache

The larger server kernel and compiled objects will grab space from data cache, affecting performance, if you do not increase total memory resources.

This section explains memory and discusses ways of determining how much more memory you need.

Understanding Memory

Read this section if you want an overview of the areas of memory, or use it for reference later.

Space in your machine's physical memory (RAM) is apportioned roughly as depicted in Figure 6-1.

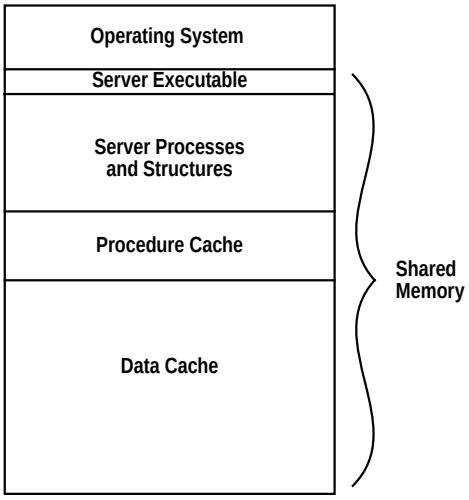

Figure 6-1: Areas of memory

The areas and their functions are:

Operating system. This part of the physical space is used by the operating system and other non-Sybase processes running on the machine.

Shared memory. This is rest of the physical memory, which is allocated to Adaptive Server by the operating system.

Server processes and structures. This area includes the **dataserver** program and the server kernel, which manages Adaptive Server. This area also includes space for structures like locks and user connections. These processes and structures take as much of the shared memory as they need.

Procedure cache. This area is used by compiled objects that execute, such as stored procedures, triggers, and views. Procedure cache grabs a set percentage (the **procedure cache percent** set by **sp_configure**) of the remaining shared memory.

Data cache. This area is where pages of data are kept in memory. In general, the more data pages that are kept in memory, the less disk

I/O the server has to do and the faster it performs. However, increasing the data cache size past an optimum amount leads to diminishing returns and wasted space.

➤ ***Note***

This outline of memory does not include other Sybase products that may be running on the same machine. You need to take these into account when planning to increase memory. Use the calculations in Chapter 3, "Analyze: Documenting Your Environment" in the section "Physical Memory Utilization" or check the installation guide for minimum RAM requirements.

Increasing Memory: Three Estimating Techniques

You can use any combination of the techniques described below to estimate your increased memory needs. A certain amount of experimenting may be necessary.

1. Calculate Data Cache Based on Error Logs

This method of comparing memory usage between old and new servers can be used when you have upgraded a test or development system. It requires that you have error logs from both the earlier SQL Server and Adaptive Server.

Your 4.9.x or 10.x SQL Server error log contains lines similar to the following:

```
Number of proc buffers allocated: 556
Number of blocks left for proc headers: 629
Number of buffers in buffer cache: 2072
```

After 11.0.x, the line, "Number of buffers in buffer cache," was changed to this:

```
Memory allocated for the default data cache: 4144Kb
```

If you have named caches in 11.0.x, each one has a similar line.

The following table shows you how to interpret these entries:

Table 6-5: Interpreting size of memory structures

Structure	What it is	Units
`proc buffers`	Contains information about a compiled object. There is one proc buffer for each instance of an object in memory. The number of proc buffers is the maximum number of procedures that can be in memory at one time.	76 bytes each
`proc headers`	Contains the entire compiled object. A large object may span more than one proc header. Larger objects means that fewer proc headers are available.	number of 2K pages
`buffers in buffer cache`	Hold data pages read into memory. Pre-11.0, all buffers are 2K.	number of 2K pages
`default data cache and named caches`	Post-11.0, multiple data caches may have buffers of 2, 4, 8, and 16K (twice this size on Stratus)	KB

Keep a copy of the error log from the earlier server. After you perform an upgrade on a test server, check the same three lines in the 11.5 error log. If you have added no memory at this point, the larger server structures will have caused the amount of cache to decrease.

Convert the sizes of proc buffers, proc headers, and data cache buffers in old and new error logs to megabytes and subtract the new from the old. This is how much you need to increase memory to keep the same amount of data cache.

2. Calculate the Growth of Objects in Procedure Cache

This method of comparing memory usage between old and new servers can be used when you have upgraded a test or development system.

In both versions of the server, run commands like the following:

```
1> use sybsystemprocs
2> go

1> select object_id("sp_help")
2> go

1> dbcc prodbuf(sybsystemprocs, <object_id>, 0
2> go
```

Use *master* instead of *sybsystemprocs* to run these commands on a 4.x server.

This **dbcc** command reports the number of memory pages used for the stored procedure whose object ID you enter. Compare results on the old and new systems. Use these results to estimate the procedure cache size needed to accommodate the objects such as triggers, views, and stored procedures that you want to hold in cache simultaneously.

3. The Simple Way: Increase by Percentage

If you want to forgo calculating present and future memory use, the following chart gives a rough estimate for increasing memory to maintain approximately the same server performance you are now getting.

Table 6-6: Percent memory increase

From Release...	Increase Memory by...
4.x	125%
10.x	50%
11.0.x	25%

How to Increase Memory

The precise steps you take to increase memory will depend your needs and environment. To increase the amount of space available to Adaptive Server, do some or all of the following:

- Increase the amount of RAM in the machine
- Increase the amount of shared memory allocated by the operating system to Adaptive Server. See the installation guide for information about increasing shared memory.
- Increase the **total memory** parameter using **sp_configure** (not to exceed the amount of memory allocated to Adaptive Server by the operating system)
- Adjust the **procedure cache percent** as necessary to produce the right amount of procedure and data cache. The **procedure cache percent** determines how much of the memory not used by the kernel goes to procedure cache. The remainder goes to data cache. This is the only way to adjust data cache.

➤ ***Note***

When you increase **total memory**, you probably don't need to keep the same **procedure cache percent** value. A larger data cache can give you performance improvements, but increasing procedure cache in the same proportion may not improve performance and can waste space. If you double your memory and your current **procedure cache percent** is 20, you can try changing it to 10 or 15 for the same performance.

See the *System Administration Guide* for a more complete discussion of memory allocation, setting the **total memory**, **procedure cache percent**, and other configuration parameters, such as **number of locks** and **open databases**, that affect memory. See the *Reference Manual* for **sp_configure** syntax.

Additional Devices and Databases

Adaptive Server 11.5 has new features that require their own databases.

sybsystemdb

A new database, *sybsystemdb*, used for two-phase commit, is automatically created during installation or upgrade. This database has a default size of 5Mb.

If you upgraded your existing two-phase commit configuration to the new 11.5 configuration, make sure that system administration scripts reference *sybsystemdb..spt_committab* instead of *master..spt_committab*.

sybsystemprocs

If you are upgrading from 4.x, skip this section and follow the instructions in the installation guide.

The system stored procedure database, *sybsystemprocs*, which was introduced in version 10 to move the stored procedures from the master device to their own device, has grown considerably from its original size. Currently, the stored procedure database is about 60 megabytes.

The increase in size requires you to extend the database. If your current device isn't large enough and you create a new device on

which to extend the database, be sure to extend the log segment onto the new device. When you extend a database to a separate device with the **alter database** command, the command extends only the data segment, not the log segment. This can cause out-of-space errors for the log.

See the installation guide for more information about this database. See also the Upgrade topic in the Technical Information Library for stored procedure database issues.

Optional Space Considerations

A number of processing changes between 11.0.x and 11.5 which contribute to improved performance require additional resources. These changes include true parallel processing on partitioned tables, new index scans, and improved prefetch. As a result, you will probably need to add resources to the Adaptive Server environment to maintain the same performance.

Partitioned Tables and Parallel Processing

Adaptive Server now supports true parallel query processing. You may want to plan to place your tables differently on Adaptive Server 11.5 to take advantage of this feature, particularly if you use partitioned tables already. While you do not have to implement this now, you may want to plan for additional devices.

Here are some considerations for planning table-partition-to-device placement:

Heavy I/O. In an environment where many users are accessing a table to perform inserts, deletes, and queries, I/O is the performance bottleneck. Place your table partitions on multiple devices with different controllers so that I/O, as well as processing, can be performed in parallel. For example, if you have four available devices and you place four tables partitions on them, you should get an increase in speed of almost four times, assuming no other factors are affecting performance.

High concurrency for inserts. If the limiting factor on performance is contention for locks while performing inserts, you could partition a heap table into many partitions, so that each would have its own last page of the page chain on which to perform inserts. For example, if you had four available devices, you could place ten partitions on each, for 40 total partitions. Inserts would wait for I/O, but they wouldn't block each other for locks.

➤ ***Note***

Parallel processing of inserts is not currently supported.

See the *Performance and Tuning Guide* for in-depth information about parallel processing.

dbcc checkstorage Disk Space and Memory

A new **dbcc** function, **dbcc checkstorage**, provides the functionality of **dbcc checktable** and **checkcatalog**, without some of the drawbacks, such as poor performance and occasional spurious errors, of these earlier commands. Most system administrators will want to use the new tool.

dbcc checkstorage requires resources not needed by earlier **dbccs**. A new system stored procedure, **sp_plan_dbccdb**, is provided to estimate the required size of *dbccdb* and memory resources, as well as suggesting suitable devices. You should run this stored procedure before creating the database to get the most meaningful results. The information provided in the following sections is to help you estimate your resource needs for planning purposes.

◆ ***WARNING!***

The sizes given here are approximate and are for reference only.

Disk Space

The database *dbccdb* should be placed on its own physical device and never on the master device.

The size of *dbccdb* depends on the size of the scan and text workspace stored within it. Space requirements for the workspaces are approximately:

- Scan — 1.2% of the target database size
- Text—25% of scan size

The minimum size of *dbccdb* is 4MB, including a 2K log placed on a separate device. You can increase database size if you want to store fault and statistics data from more than five runs of **dbcc checkstorage** or run the procedure on more than two user databases at the same time.

Memory

You need a dedicated named cache for **dbcc checkstorage** and it must have a 16K buffer pool. The size of the named cache used should be 20% of total workspace size or 640K per worker process (for parallel **dbcc**).

You have to configure a maximum number of worker processes for *dbccdb* for each target database. These worker processes allow **dbcc checkstorage** to perform its checks in parallel, improving performance. For each target database, you should configure at least as many worker processes as there are devices for the target database.

See the *System Administration Guide* for information about **dbccdb** requirements and the *Performance and Tuning Guide* for information about named caches and worker processes.

7 Test: Ensuring Stability and Performance

Introduction

This chapter will help you evaluate testing methods and develop a testing plan. It contains these sections:

➤ ***Note***

This chapter is intended as a guide to various kinds of tests. You should use those suggestions that fit your business needs and resources. Full-scale testing as described here is recommended but not required for migration success.

The Goal of Testing

The primary goal of testing is to ensure that after migration:

- Application behavior is predictable.
- Application and operational service levels are preserved or exceeded.
- The test and production systems are stable and the data is safe.
- The upgrade is successful and does not adversely impact the production system.

Setting Up the Test Environment

Ideally, you should set up a dedicated hardware configuration (including subnets) and SQL Server exactly like your production system. Creating an identical system lets you make valid comparison, perform real tuning as part of migration effort, and if you wish to do so, switch the test system to production later on.

Make Backups

Make backups of the production system. You can use these to populate your test system and to restore it when necessary.

Use Scripts to Create the Test System

Using the object creation scripts you gathered, wrote or reverse engineered in Chapter 3, "Analyze: Documenting Your Environment", build a test environment matching your production system.

Use backups or the **bcp** scripts to populate your test databases.

➤ ***Note***

When you create a new database and then load **bcp** files, you reduce the fragmentation that may be present in the production system and change the performance characteristics of the database. If this is not desirable because you do not intend to rebuild your production environment, you may prefer to create your databases as described in the next section, "Create Your Databases by Loading Backups".

Run **dbcc**'s in the test databases immediately to be sure that there are no problems at this stage.

Create Your Databases by Loading Backups

If you do not intend to rebuild your production environment, you may prefer to create your test databases with the **for load** option of the **create database** command. This makes your test databases more representative of the current production environment in terms of fragmentation and density.

Follow these steps:

1. Create the database with the **for load** option of the **create database** command.
2. Load the backups you made of the production database.
3. Issue the **online database** command. This command automatically upgrades the database if it is not at the 11.5 level.

The **online database** command was added in SQL Server 11.0. For command syntax, see the *Reference Manual*.

If the Test Environment Is Not an Exact Duplicate

If you have to use a smaller system for testing, try to have components identical to the production system, such as operating system level, drivers, and disk types.

You have to make adjustments when you have less disk space or memory in the test system. Scale down databases proportionally, while retaining the same data distributions so optimizer decisions remain constant. Scale down memory to ensure consistent I/O rates.

Reproduce data layout across available devices as closely as possible.

If you use fewer CPUs, adjust transaction arrival rates (that is, load) and concurrent users proportionally.

Prioritizing Applications to be Tested

Since it may be difficult to test all application functions, gather user input to identify the most critical transactions. Write tests for those functions using the techniques described in the following sections. Functions not tested at this time can be validated during user acceptance tests.

Establishing Performance Criteria

Depending on the migration plan you developed, you may want to meet or exceed the performance you get with your current system. For example, you might decide to apply guidelines like these:

- For **parallel with replication**:
  - Be sure to measure the overhead of the replication mechanism. For example, if the replication costs 10%, do not begin parallel operations until release 11.5 is tuned to outperform by 10%.

  - For an around-the-clock operation, you may decide that an initial goal of breaking even is reasonable.
- For **cutover without replication**, gear the migration for equivalent performance between the old and new systems. A goal of breaking even the first week is reasonable.
- A **phased cutover** is subject to the highest performance expectations. Some performance tuning of the production workload after cutover of the production server may be best. You can time the production server cutover to occur as soon as performance gains are acceptable and testing is successful.

See Chapter 4, "Prepare: Writing a Plan and Getting Ready to Migrate" for information on migration methods.

Developing Fallback Procedures

Before you begin testing, be sure that you know how you will return the test system to a known state when you have a problem. You will use the same fallback plan when you migrate the production system.

During simple testing cycles, it's sometimes faster to back out unwanted changes. However, this is not recommended for **timed runs** during benchmarking. You have to restore from backup after each timed run to return the system to a known state.

Back up all databases before and after the test system upgrade, as you would for a "real" upgrade. The backups preserve the layout of data on disk and help you avoid confusion due to fragmentation and page splits.

To ensure source code control, use scripts for all changes to objects. This makes it much easier to recreate your environment if necessary.

Summary of Testing Techniques

You can use a variety of testing techniques in your test plan. The following table describes the advantages and disadvantages of various testing techniques and tools, including:

- Ad-hoc testing
- Manual test scripts and cases
- Keystroke capture tools
- Transaction generators

- Production load capture

Table 7-1: Summary of testing techniques

Technique	Description	Advantages	Disadvantages
Ad hoc testing	Manually walk through important application processes, screens, and reports.	• Easy to implement. • Tests front-end applications and back-end servers.	• For complex applications, code coverage is too small. • Difficult to distinguish front-end and back-end bottlenecks if response time is a determining factor. • Impossible to obtain production multiuser load, which misses concurrency and capacity issues altogether.
Manual performance scripts and cases	Specify input and compare with known outputs.	• Easy to implement. • Common basis for regression test suites.	• Only tests back- end server. • Back-end focus may help locate the cause of a problem. • Impossible to obtain production multiuser load, which misses concurrency and capacity issues altogether. • No ad hoc query testing. • Depends on strong analysis of process or transaction profiles.
Keystroke capture	Record and replay keystrokes and mouse clicks into an application.	• Tests front-end applications and back-end servers. • Tool may include powerful language and looping capabilities to manipulate inputs for multiuser concurrency and capacity testing	• Heavy processing requirements. May require additional hardware. • May increase development time for creating multiuser test simulations, and add time for debugging test harness. • Depends on strong analysis of process or transaction profiles.

Table 7-1: Summary of testing techniques

Technique	Description	Advantages	Disadvantages
Concurrency and capacity testing	Using tools like Cyrano Workbench and Impact, Performix Empower, Mercury Interactives, LoadRunner	• Tests both front and back ends • Language and looping constructs make these incredibly powerful tools	• Heavy processing requirements for test tool—may even require additional hardware or else results could be skewed • Learning and development curve to write multi-user test simulations • Risk of bugs in test harness could skew results • Dependent on strong analysis of process / transaction profiles
Transaction generation	Thin Client to Simulate user execution of transactions	• Strong multiuser load testing • Focus on back-end server issues	• May increase development time for creating multiuser test simulations, though learning and development curve generally less than keystroke capture tools • Adds time for debugging test harness to prevent skewed results • Depends on strong analysis of process or transaction profiles
Production load capture	Using tools to capture real transactions in a production environment, including performance and semantic characteristics, and resubmit in a test environment for analysis	• Tests real production loads, including ad hoc queries • Especially useful when little or no analysis of transaction profiles is available	• Introduces new software into a production environment • Production and test system configurations must be identical for valid performance analysis

Writing Performance Scripts

This section discusses the basics of writing performance scripts.

Write Benchmark Scripts

In general, you have to write special benchmark scripts rather than rewriting applications as benchmarks.

To write the benchmark script:

- Add function to *funcs.c* for each transaction
- Generate any run-time data required (such as primary key to select or data to insert)
- Write code to submit SQL or stored procedure to Adaptive Server (for example, using **dbsqlexec()** calls)
- Write code to process result set(s)
- Name each transaction explicitly for (for example, "**begin tran cust_update**") to make it easier to identify in system procedures and tables
- For a stored procedure-based system, verify the parameters that make the stored procedures work. If the parameters need to vary for a meaningful test, add the necessary logic.

Volume is critical in performance simulation. A script roughly equivalent to an application, running at the normal production volume for that application, is usually better than a script functionally matched to an application, but running at only half the volume.

If workload is based on client PCs issuing Transact-SQL, you can use performance monitoring tools, available in the market, to capture data streams.

Drivers

In addition, you need drivers for these tasks:

- Error handling
- Deadlock handling
- Result handling
- Timing measurement
- Run-time data generation

General Error Handling

In the event of an error, you can either throw away the transaction and not count it or, depending on the requirements of the test, restart the transaction and count the entire response time.

Deadlock Handling

You have to resubmit deadlocked transactions to get a realistic time measurement. You can count average response time with and without deadlocks.

Result Handling

You can capture query results by fetching the entire result set back to the client and binding it to local variables. Outputting results to a file may increase your time by requiring additional file I/O and operating system buffering.

Time Measurement

Time measurement is especially important in multi-tiered applications where a bottleneck could occur at any level. Generally, you should time database transactions, not business functions. Start at transaction "send" and stop at the last row processed in the result set. Logical business operations can be aggregated later.

Granularity is important for problem identification and resolution. Important measurements to take include:

- Throughput (number of transactions per second / minute / hour)
- Average and maximum response time by transaction
- A histogram of response time ranges by transaction. For example:
  - Less than 1 second
  - Between 1 and 2 seconds
  - Between 2 and 3 seconds
  - Over 3 seconds

Runtime Data Generation

When using runtime-generated data, skewed keys may be a problem. Selects may show an unrealistically high cache hit rate, and inserts, updates, and deletes may appear to have concurrency problems. To avoid these problems, use the entire range of key values in your tests, unless you are trying to recreate a particular business situation.

Using a separate file for each transaction can prevent skewing; however, it requires additional code to sync users. Using a separate file for each user is more work to create but also prevents skewing

Using a memory generator works well to prevent skewing and makes the test easy to administer. However, the benchmark is then not 100% repeatable.

Test Cycle: Summary of Tests

This section summarizes a complete test cycle, with tests that target specific issues, including old and new functionality, performance under multi-user loads, integration, and user acceptance.

Table 7-2: Stages of Testing

Stage	Purpose	Best Technique
Functional testing	For each application or process, addresses the following questions: • Are there any obvious bugs? • Do transactions return the same results? • Will the application break anywhere? If you decide to use new functionality: • Does the release provide the functionality it markets? • What are the limitations of the new features of this release?	Single-user: • Ad hoc test • Manual test scripts and cases • Existing application test suites
Stress testing (benchmarking)	Using very heavy loads, addresses the following questions: • Are there any bugs related to multi-user loads? • Is the performance of critical transactions as good or better? • Is the new release stable under load?	Multiuser: • Keystroke capture • Transaction generator • Production load capture

Table 7-2: Stages (continued) of Testing

Stage	Purpose	Best Technique
Integration testing	Ensure that all system components work well together, such as: • Batch processing • Online transaction processing (OLTP) • Decision Support Systems (DSS) and ad hoc query • Operations, including backup, recovery, and **dbcc** commands • Sybase products other than SQL Server • Third party products	Test suite models all system components
End-user acceptance testing	Execute acceptance tests specific to the environment. Also cover functions not prioritized into earlier stages. **Note:** The other stages, done well, should have caught most of the problems.	Standard acceptance tests
Final migration plan testing	• Ensure that you are fully prepared by walking through the upgrade / migration plan. • Verify that fallback procedures work. • Identify and test the contingencies.	Walk through every upgrade step, including fallback strategies

➤ ***Note***

Sybase recommends the use of Cyrano migration products for testing.

Test Cycle: Testing for Performance

This section expands on performance benchmarking before and after upgrade. For a summary of the entire test cycle, see,"Test Cycle: Summary of Tests" on page 7-9.

This section covers the following tasks:

➤ ***Note***

Benchmarks test server processing speed. Therefore, run your benchmarks from a light-weight client so that all processing happens at the back-end.

Pre-Upgrade Single-User Tests

After you create the test system at the same SQL Server level as the production system and before running benchmarks, you need to sync the test system with the production system so that comparisons to the new server are valid.

Optimizer

Compare **showplan** and output from **dbcc 310** and **302** which you gathered in Chapter 2, "Analyze: Documenting Business Requirements", to that of the test system to verify that the optimizer is making the same decisions. Some possible causes of optimizer differences include:

- Data and data distribution problems. Check data layout and be sure that these match the productions system.
- Index definitions. Verify that your reindexing scripts worked. Using sp_helpindex, compare the index definitions of the production and test systems.

Resolve any problems before going on.

I/O

Verify that the test server is performing the same amount of I/O. Physical and logical I/O should be in the same proportion as in the production system. Use **statistics io** output to compare.

Pre-Upgrade Multi-User Tests

When the optimizer is behaving the same as it did in the production system, you are ready to run benchmarks.

Untimed Benchmarks

In multi-user mode, run untimed benchmarks as often as necessary, capturing response time and throughput metrics. Resolve:

- Differences, such as cache hit rate, to the production system
- Bottlenecks caused by saturated devices
- Other problems or mistakes

Between runs, restore your databases from backup or perform a quick reset.

Timed Benchmarks

Always restore your databases from backup so they are in a known state when running timed benchmarks. Gather response time and throughput metrics. Run as often as necessary to fix problems and be able to reproduce the final results.

Test System Upgrade

Perform the upgrade on the test system following the instructions in the installation guide, *Installing Sybase Products*. Be sure to perform all steps so that the test upgrade is a walkthrough of the real upgrade. Make the memory and disk space changes you determined that you needed in Chapter 6, "Implement: Making Required Database Administration Changes"**.** However, do not change your Adaptive Server configuration to use new features at this time.

➤ ***Note***

Your first objective after upgrade is to test Adaptive Server's out-of-the-box performance. Wait until later to try new performance tuning features.

Resolve all problems and note them for reference. The installation guide gives detailed troubleshooting information.

After the upgrade:

- Perform **dbcc**'s on all databases.
- Back up all databases, including *master* and *sybsystemprocs*. You will need backups to restore the test system to a known state between timed benchmarks.

Post-Upgrade Single-User Tests

In single-user tests, verify that:

- The optimizer is creating the same or better query plans. Differences in query plan may be due to optimizer changes, depending on the SQL Server version you were running. Determine whether you need to change application code to compensate. See Chapter 7, “Test: Ensuring Stability and Performance”, for possible optimizer issues.
- I/O counts, physical and logical, are the same as they were before the upgrade.

Post-Upgrade Multi-User Tests

After reloading backups to return the test system to a known state, run untimed and timed benchmarks.

Untimed Benchmarks

In multi-user mode, run untimed benchmarks as often as necessary, capturing response time and throughput metrics. Resolve:

- Differences, such as cache hit rate, to the pre-upgrade state
- Bottlenecks
- Other problems or mistakes

Be sure that you have enough default data cache after the upgrade. Tune as needed to fix problems. Between runs, restore your databases from backup or perform a quick reset.

Refer to the *Performance and Tuning Guide* for more information. See also Technical Information Library for the latest TechNotes and White papers on issues that may affect performance.

Timed Benchmarks

Always restore your databases from backup so they are in a known state when running timed benchmarks. Gather response time and throughput metrics.

Tune as necessary to achieve your performance criteria.

Run tests as often as necessary to fix problems and be able to reproduce the final results.

A The Cyrano Migration Pack

What is the Cyrano Migration Pack?

The Cyrano Migration Pack (CMP) is a comprehensive and integrated range of industry leading tools for accelerating, securing and certifying any migration from Sybase SQL Server 4.x, 10.x, or 11.0.x to Sybase Adaptive Server Enterprise 11.5. Cyrano Migration Pack is fully available today. These tools were developed to reduce the risk involved in the migration and assist the user in identifying application and database issues during the migration process to be able to take advantage of the performance characteristics of Adaptive Server 11.5.

Cyrano is the preferred methodology and tool set used by Sybase Professional Services when engaging in a migration consulting engagement.

Who Should Use the CYRANO Migration Pack?

Businesses that can benefit from Cyrano Migration Pack include:

- Existing users of Sybase SQL Server 4.x, System 10, or System 11 migrating to Sybase Adaptive Server Enterprise 11.5
- Existing post-migration users of Adaptive Server 11.5 who want to fine tune and certify their migration
- Organizations with medium to large scale Sybase applications
- Organizations running any application regardless of the client development tool or language used. (Additional benefits are possible if the application was developed with PowerBuilder.)

Features of the Cyrano Migration Pack

The Cyrano Migration Pack...

- Delivers a complete suite of tools for managing and controlling the Sybase application migration process
- Provides SQL statement unit testing, new SQL access strategy validation, and SQL performance optimization
- Provides automatic comparison between results sets from both releases, and pinpoints the differences

- Provides deployment testing with an explanation of unacceptable response times by identification of faulty transactions
- Lets you reuse Cyrano WorkBench and Cyrano Production after Migration for development, testing and monitoring application databases on Adaptive Server 115.
- Supports any Sybase migration strategy and methodology

Benefits of the Cyrano Migration Pack

The Cyrano Migration Pack...

- Has tools to support any Sybase methodology
- Requires only one product set to support the whole process— one stop shopping
- Can accelerate, secure and certify the migration
- Confidently moves Adaptive Server into production and takes full advantage of its power
- Ensures migration quality and application performance tuning
- Simplifies and shortens any future migration to new version of Adaptive Server by saving and playing back scenarios using Cyrano ImpactKey

Components of the Cyrano Migration Pack

The following tools comprise the Cyrano Migration Pack:

- **Cyrano WorkBench Server and Client**

  Identifies SQL and offers the migration specialist the opportunity to change application SQL that may not be performing up to standards.

- **Cyrano Production Server and Client**

  Monitors the production databases after migration.

- **Cyrano Xtract (GenDDL)**

  Extracts all relevant SQL definition text from a SQL Server DBMS.

- **Cyrano Xplain**

Looks for new reserved words, optimizer strategy issues, correlation names, subqueries, column headings, and SQL ANSI comments in your scripts or databases.

- **Cyrano Xact(ResDiff**)

  Compares results set differences between Pre-11.5 releases of SQL Server and Adaptive Server 11.5.

- **Cyrano Xamine(TechDiff) and Xpose**

  Compares performance differences in logical I/Os and client response time between Pre-11.5 releases of SQL Server and Adaptive Server 11.5.

For More Information

For more information, see Cyrano's home page at http://cyrano.com. Also see the Sybase Migration Resource Guide at http://www.sybase.com/migration.

B Worksheets for Your Current Environment

This appendix provides guidelines and sample worksheets for gathering information that will help you in planning for migration. The topics are:

As part of migration planning, see the Sybase Migration Resource Guide at http://www.sybase.com/migration for current offerings.

SQL Server Operational Worksheets

The following worksheets are for gathering business requirements. For more information, see Chapter 2, “Analyze: Documenting Business Requirements”.

This worksheet includes the following sections:

Operational Business Requirements

General operational information can help with documentation and success criteria.

Table B-1: Operational requirements

Database Name	Operational Hours	Maximum Downtime	Comments

Backup and Restore Procedures

This worksheet is helpful for a general survey of your backup and restore procedures.

Table B-2: Backup/restore procedures

Task	Yes/No	Comments
Are backup and recovery procedures documented?		
Are automated dump procedures in place?		
What is the number of generations of dumps?		
Are dumps kept offsite?		
Are maintenance activities documented?		
What is the frequency of SQL Server error log scanning?		
Is database space utilization monitored?		
Has database capacity planning been performed recently?		

Database Dump Details

Detailed backup and restore information is helpful for defining success criteria.

Table B-3: Backup/restore details

Database Name	Frequency of Database Dumps	Dump Device Used	Frequency of Transaction Log Dumps	Dump Device Used	Comments

Maintenance Procedure Details

Detailed maintenance information is helpful for defining success criteria.

Table B-4: Maintenance

Database/Object Name	**Frequency of** *dbcc checkdb/checktable*	**Frequency of** *dbcc checkalloc/tablealloc*	**Frequency of** *update statistics*	**Frequency of Monitoring Space Utilization**

Data Architecture Worksheet

Documenting the flow of information is discussed in Chapter 2, “Analyze: Documenting Business Requirements”. The Data Architecture Worksheet includes these sections:

Client Application Components

Application profile and performance requirements information is helpful for defining success criteria.

Table B-5: Client applications

Application Name	Type of Application	Tool Used to Write Application	Where Client Runs	Number of Concurrent Users	DBs Accessed	Maximum Downtime (per day)	Average Response for Priority 1

Production Performance Metrics

Measure current production performance metrics, using operating system monitors, for the following:

- CPU

  Measure the average and maximum CPU utilization (aggregate and per CPU on SMP servers) per "time window" (online, batch, and so on) per server.

- Disk I/O
  - Measure I/Os per second per disk and controller, and I/O queue lengths per "time window" per server.
  - Also measure total I/Os, reads, and writes per second per Sybase device per "time window" per server.
- Concurrency

  Determine the average lock contention. You can use **sp_who** to determine the processes currently running, and **sp_lock** to find out which current processes hold locks.

- Network I/O
  - Measure the packets per second per NIC per "time window" per server.
  - Also measure the TDS packets ("sent from" and "received by") per "time window" per server.
- Memory
  - Determine the paging/swapping rates per second per "time window" per server.
  - Also determine the data and procedure cache hit rates per "time window" per server.

Transaction Profile

Repeat this section for each database on the SQL Server. Application processing information at the transaction level is helpful for documentation and success criteria.

Table B-6: Transaction profile

Process (Xact) Name	Process Type (OLTP, batch)	Xact Priority	Frequency per User (per hour)	Source Code Type (stored procedure, ESQL, etc.)	Average Response Time Required	Maximum Response Time Required	Current Average and Maximum Response Time

➤ ***Note***

To quickly identify response time problems, save **showplan** output for all critical transactions.

SQL Server Infrastructure Worksheets

Documenting your environment is discusses in Chapter 4, “Prepare: Writing a Plan and Getting Ready to Migrate”. The infrastructure worksheets include:

Record this information for both the **production** and **development** environments.

Host Configuration

This section provides worksheets for host configuration.

Hardware

Record technical support information about your hardware manufacturer:

Table B-7: Host hardware

SQL Server Machine
Make:
Model:
Customer ID with hardware vendor:
Technical support phone number:
Technical support hours:

Table B-7: Host hardware

SQL Server Machine
Technical account manager: • Name • Phone number (s) • Pager number

Record CPU resources:.

Table B-8: CPU resources

CPU
Number of physical processors:
Chip speed:
Number of processors available to SQL Server:
Other CPU-intensive processes / threads:
Processes / threads bound to specific CPUs:

Table B-8: CPU resources

CPU
Processes / threads run at high priority:

Physical Memory Usage

List all the major processes and memory requirements running on each server. See the formulas in *Table 3-7: Server memory map*.

Table B-9: Runtime memory usage

Application	Runtime Memory Usage
Operating system	
Networking software	
SQL Server memory parameter	

Table B-9: Runtime memory usage

Application	Runtime Memory Usage
Open Servers Include: • Backup Server • sybmultbuf • Replication Server • Monitor Server • Gateways (list)	
Other applications	
Total memory required	
Total memory installed	

Disk I/O Configuration

General disk information can help with firmware incompatibilities and capacity planning.

Table B-10: Disk I/O

Controller Number	Make and Model	Firmware Revision	Months in Service	Transfer Rate (KB/sec)

The following disk layout information can help with firmware incompatibilities, nearing Mean-time Between Failures (MTBF), load balancing, and capacity planning.

Table B-11: Disk and firmware use

Physical Device Name	Make and Model	Firmware Revision	Months in Service	Controller Number	Capacity (MB)	Through-put (I/Os per second)	Transfer Rate (KB per second)

The following disk layout information can help in case redistribution is required, load balancing, and capacity planning.

Table B-12: Disk and partitions

Physical Device Name	Partition Number	Used by (Sybase, UNIX File System, etc.)	Device Name	OS Mirrored Device Name	Capacity (MB)	Cylinder Range

The following logical volume information can help in case redistribution is required, load balancing, and capacity planning.

Table B-13: Logical volumes

Logical Volume Device	Member Disk Partitions	Used by (Sybase, UNIX File System, etc.)	Sybase Device	Mirror Logical Device	Capacity (MB)	Stripe Width (MB)

Network Configuration

Network layout information can help with firmware incompatibilities, MTBF, and capacity planning.

Table B-14: Network layout

Physical Device Name	Make and Model	Firmware Revision	Months in Service	Supported Protocols	Network Address	Transfer Rate (KB/second)

Tape Configuration

Tape layout information can help with firmware incompatibilities, MTBF, and capacity planning.

Table B-15: Tape layout

Physical Device Name	Make and Model	Firmware Revision	Months in Service	Capacity (MB)	Controller Number	Transfer Rate (KB/second)

Operating System Configuration

Detail the operating system.

Table B-16: OS details

Server Hardware
Name:
Release level:
Patch history:
Kernel configuration parameters:
Swap space size:
Technical support phone number:
Technical support hours:
Technical account manager: • Name • Phone number(s) • Pager number

SQL Server Configuration

Document general information about the SQL Server configuration.

Table B-17: SQL Server configuration

General Configuration
Home directory:
Components, release and fix levels:
Locations / names of scripts to rebuild database environment:
sp_configure configuration values:
buildmaster configuration values:

Run **dbcc memusage** on SQL Server during an off-peak time or in single-user mode.

Table B-18: dbcc output

Usage	MB	2K Pages	Bytes
Configured memory			
Code size			
Kernel structures			
Server structures			
Page cache			
Proc buffers			
Proc headers			
Number of page buffers			
Number of proc buffers			

Database Devices

Database device information can help in case redistribution is required, load balancing, and capacity planning.

Table B-19: Database devices

Database Device Name	Physical Device Name	Virtual Device Number	Capacity (2K Pages)	Mirrored Device Name

Databases and Segments

Database and segment information can help with load balancing and capacity planning.

Table B-20: Databases and segments

Database Name	Database Options Set	Fragment (page range)	Size (in MB)	Segment Names	Device Name

Dump Devices

Dump device information can help with load balancing and capacity planning.

Table B-21: Dump devices

Database Device Name	Physical Device Name	Media Type	Capacity (MB)

C Sample Migration Plan Outline

The following is a high-level migration plan produced by Sybase Professional Services for a Sybase customer. The plan is for a migration from SQL Server 4.9.2 to SQL Server 11.0.x and reflects site-specific needs.

The plan touches on technical issues specific to an 11.0.2 migration and does not address technical issues particular to Adaptive Server 11.5. For information on technical changes in Adaptive Server 11.5, see Chapter 5, “Implement: Making Required Application Changes” and Chapter 6, “Implement: Making Required Database Administration Changes”.

➤ ***Note***

This plan is intended to be an example only. Your own plan may differ significantly depending on your needs and resources.

Migration Approach Outline

The SQL Servers to be migrated at Acme Financial are now at version 4.9.2. Before version 4.9.2 database objects can be migrated to System 11, changes need to be made to make them compatible with System 11. Our approach is to try to automate as many as possible of the changes needed, and after that to try to automate testing of the changes as far as possible.

Changes and new features in System 10 and 11 have been documented in several official Sybase publications. In the current document, these changes have been divided into several phases. Each phase will be handled differently during the migration process.

To help automate changes, a program will be written in the PERL language. Some changes will be identified and made by the PERL program. Other more complex changes will only be identified but not made by the PERL program, because manual intervention will be needed to decide whether and how to make these changes.

After all of the identified changes have been made, scripts will be compiled in SQL Server. We expect to find more syntax errors because some issues, such as correlation name handling, are too complex to be picked up in the PERL program. Compilation errors will be iteratively fixed until all objects compile cleanly.

After the successful compilation of all database objects, a Sybase-supplied system stored procedure will be run to identify changes due to subquery processing. Differences in SQL Server 11 can cause unexpected result to be returned by queries.These changes will be made manually after identification.

Finally, some changes will neither be identified nor fixed during this phase of the migration process. These unfixed changes, which are not detected by the parser, may produce run-time errors, for example if existing programs rely on the specific wording of an error message which has been changed. These are documented here for information purposes only and to help diagnose future error conditions.

New System 11 features, such as new table constraints, will not be introduced into during the migration process to minimize the possibility of error conditions.

The following sections define phases for upgrading the Acme Financial's trading servers.

PHASE 1: Issues to be Addressed by PERL Program

The following issues will be identified and fixed automatically by the PERL program:

- Identify reserved words and change to agreed upon option
- Identify double hyphens (--) and change to -(-))

PHASE 2: IssUes Identified But Not Fixed by PERL Program

The following issues will be identified by the PERL program, then fixed manually:

- Identify all instances of **between** in **where** clauses in SQL code
- Identify **set arithabort** and **set arithignore** options
- Identify instances of:
  - **in, any, not in**
  - **or** with **exists, >all, <all,** or **exists,**
  - **select distinct**
  - NULL results.
- Identify subqueries that use Transact-SQL functions in the **select** statement.

Example:

```
update tableA set field1 =
(select max(field1) from tableB TB
where tableA.field = TB.field1)
```

- Identify subqueries using **distinct** / **in** / **not** / **exists** / **any** clauses.

  Example 1:

```
select count(*) from publishers
where exists (select * from titles
where titles.pub_id = publishers.pub_id)
```

  Example 2:

```
select pub_name from publishers
where pub_id in
(select distinct pub_id from titles
where titles.pub_id = publishers.pub_id)
```

PHASE 3: Issues Identified by Object Compilation in System 11

After upgrade, the following objects will be created in sequence: tables, indexes, user stored procedures, triggers, views, users, and permissions. We will then identify and address any issues found by the SQL Server parser:

- Self-join format (repeated table names) must include table aliases.
- Identify NULL column headings in select into using aggregates (**avg**, **min**, **max**, **sum**, **count**)
- Defined table aliases in queries must be used for column references (also called correlation name handling)

PHASE 4: Manual Analysis and Resulting Changes to Object Code

After all objects have been successfully created on System 11, the following issues need to be identified and documented by Sybase staff. Testing and changes will be performed by Acme Financial staff to ensure similar behavior between current and new objects:

- Identify objects with subqueries by using **sp_procqmode** against database

PHASE 5: Other Issues Not Checked For or Changed

Some issues will not be detected automatically or manually by Sybase staff. Some of these issues may produce run-time errors under certain circumstances:

- New and changed error messages issued by Server
- New numeric datatype
- Additional digits of precision for approx-numerics (display formats)
- New datatype hierarchy changes
- New behavior of conversion of datatype changes
- New overflow behavior for some datatypes
- New output for system stored procedures
- Changes to system tables
- NULL passwords no longer allowed
- Database administration-specific changes

D Sample Migration Task Lists

In This Appendix

This appendix provides the following samples:

Sample Task List Template

The following table is a suggested format for detailing the tasks you need to perform as part of your migration. Modify it as necessary:

Task	Date	Begin Time	End Time	Duration	Owner	Status

General Migration Task List Example

The following general task list shows typical migration tasks, with differing details for the install/load approach (install an new server, then load compatible dumps from earlier server) and one using the upgrade approach (run the upgrade program on the current databases). Your own migration task lists may differ in detail and order.

See Chapter 4, “Prepare: Writing a Plan and Getting Ready to Migrate” for more information on migration planning.

Migration Analysis

The following tasks are related to analyzing the environment:

Document Current Configuration

1. Establish cutoff point for environment.
2. Document current server installation information.
3. Document current server configuration values.
4. Document hardware configuration.
5. Document applications per server.
6. Document application server requirements.
7. Document application client requirements.
8. Document related software and middleware.
9. Retrieve source database creation scripts.
10. Retrieve source object creation scripts.
11. Get counts of all server and database objects.
12. Review configuration document.
13. Update configuration document.

Gather Business Requirements

1. Define business requirements.
2. Define constraints.
3. Define application dependencies.
4. Define dataserver dependencies.
5. Prioritize applications.
6. Identify vendor issues.
7. Review requirements document.
8. Update requirements document.

Conduct Compatibility Analysis

1. Analyze hardware compatibility.
2. Analyze operating system compatibility.

3. Analyze other Sybase software compatibility.
4. Analyze non-Sybase software compatibility.
5. Analyze middleware / API compatibility.
6. Analyze communications compatibility.
7. Analyze client platforms compatibility.
8. Document analysis results.
9. Review compatibility analysis.
10. Update compatibility analysis.

Develop Migration Strategy

1. Draft migration strategy.
2. Review strategy (team).
3. Update strategy (team).
4. Review strategy (user / sponsor).
5. Define migration downtime impact.
6. Notify affected departments.
7. Revise implementation plan.
8. Obtain user signoff for migration and implementation.

Migration Preparation

Write Test Plans and Test Scripts

1. Write system functional tests.
2. Write integration tests.
3. Write stress tests.
4. Write user acceptance tests.
5. Review test plans.
6. Update test plans.
7. Create test scripts for each phase of testing.
8. Execute scripts on source system to establish baseline.
9. Get user signoff on test plans and baseline results.

Prepare Applications For Migration

1. Search for new reserved words.
2. Search for subqueries.
3. Verify new database conversion and computations.
4. Search for **set**, **arithabort**, and **arithignore.**
5. Search for **select into** with NULL column headings.
6. Design code changes.
7. Get user signoff on application changes.

Design and Develop Server Migration Scripts

1. Design and develop server configuration scripts.
2. Design and develop file system configuration for devices.
3. Create database device scripts.
4. Prepare security, login, and password corrections.
5. Create security scripts.

Design and Develop Database Migration Scripts

1. Create database scripts.
2. Create database object creation scripts.
3. Create database security scripts.
4. Modify / create system administration scripts.

Design and Develop Data Migration Scripts

1. Design and develop data extract scripts (such as **bcp**).
2. Determine optimal bulk copy options.
3. Design and develop data load scripts.

Perform Other Pre-migration Tasks

1. Design fallback tasks.
2. Obtain user signoff on fallback tasks.
3. Perform backups.

4. Set up source code control environment.
5. Set up new user environment.
6. Develop other migration aids.

Implement Migration (Using Install/.Load Technique)

Create Target Environment

1. Verify target system readiness.
2. Move migration scripts to target system.
3. Configure file system.
4. Complete installation / upgrade worksheet from installation guide.
5. Install Adaptive Server, Backup Server, and Open Client.

Perform Server Migration

1. Restrict access to systems and servers.
2. Create database devices.
3. Execute server security and other scripts.

Perform Database Migration

1. Execute database creation scripts
2. Create database partitions and segments.
3. Create database objects (defaults, constraints, rules, views, etc.).
4. Create database tables and stored procedures.
5. Execute database security script.

Perform Data Migration

1. Execute source data extraction scripts.
2. Move data from source to target platform.
3. Execute data load scripts.

Complete Server and Data Migration

1. Create indexes and triggers.
2. Run **dbcc** commands.
3. Dump databases.

Perform Application Migration

1. Develop application code changes.
2. Unit test application changes.
3. Analyze and correct application changes.
4. Configure applications to access new server.
5. Perform preliminary tuning.
6. Allow user access to system and server.
7. Insure no access to obsolete files.
8. Notify users of availability.

Implement Migration (Using Upgrade Technique)

Upgrade SQL Server

1. Verify target system readiness.
2. Configure file system.
3. Install software.
4. Use **sybsetup** or **Server Config** to perform upgrade.

Complete Migration

1. Run **dbcc** commands.
2. Dump databases.

Perform Application Migration

1. Develop application code changes.
2. Unit test application changes.
3. Analyze and correct application changes.

4. Configure applications to access new server.
5. Perform preliminary tuning.
6. Allow user access to system and server.
7. Insure no access to obsolete files.
8. Notify users of availability.

Migration Quality Assurance

Perform System Tests

1. Perform functional tests.
2. Compare functional test results to baseline.
3. Analyze functional test results.
4. Perform corrective actions.
5. Retest as necessary.
6. Document functional test results.
7. Obtain user signoff on functional test results.

Perform Integration Tests

1. Perform integration tests.
2. Compare integration test results to baseline.
3. Analyze integration test results.
4. Perform corrective actions.
5. Retest as necessary.
6. Document integration test results.
7. Obtain user signoff on integration test results.

Perform Stress Tests

1. Perform performance/capacity tests.
2. Compare capacity test results to baseline.
3. Analyze capacity test results.
4. Perform corrective actions.

5. Retest as necessary.
6. Document capacity test results.
7. Obtain user signoff on capacity test results.

Perform User Acceptance Tests

1. Perform acceptance tests.
2. Compare acceptance test results to baseline.
3. Analyze acceptance test results.
4. Perform corrective actions.
5. Retest as necessary.
6. Document acceptance test results.
7. Obtain user signoff on acceptance test results.

Perform Production Data Refresh

1. Schedule production data cutover.
2. Extract production data.
3. Move data to target.
4. Load data into Adaptive Server production environment.
5. Backup new environment.
6. Notify users of data refresh.

Parallel Migration Task List Example

This sample task list covers a parallel migration with replication approach to upgrade SQL Server 10.x to Adaptive Server11.5.

The scenario for this task list is:

- A major telecommunications company cannot afford system downtime for the upgrade.
- In the event of failure, the system cannot be down more than 15 minutes and cannot afford more than 1 hour of data loss. The migration must support these contingency requirements if a production back-out is required.

This example does not repeat the preparation steps to given in the previous example.

Define Test/Acceptance Criteria—Regression Test Suites

Back-end regression test suite—production loads

1. Identify back-end queries.
2. Encapsulate back-end queries.
3. Create back-end test suite (**showplan**, **stat io**, and **stat time** wrappers).

Front-end simulation regression test suite

1. Identify target user functions.
2. Capture and map SQL for target user functions.
3. Encapsulate SQL for user functions.
4. Create front-end simulated test suite (**showplan**, **stat io**, and **stat time** wrappers)

Front-end Phase 1 and 2 regression test suite

1. Identify front-end test scenarios.
2. Review front-end application and acceptance / test procedures.
3. Document functional test approach.
4. Compose front-end test mix matrix.

Set Up Target Production Environment

1. Identify physical drive configuration for Server A (production) and Server B (shadow).
2. Configure physical drives.
3. Perform a dump of the production environment.
4. Install SQL Server 10.x on Server B.
5. Configure SQL Server 10.x on Server B, to duplicate Server A.
6. Update the interfaces.
7. Create the databases.
8. Load the Server A dump on Server B.
9. Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**).

10. Synchronize user IDs.
11. Run **checkpoint.**

Set Up Replication Server

1. Install Replication Server on Server B.
2. Configure Replication Server on Server B.
3. Install Replication Server on Server A.
4. Configure Replication Server on Server A (secondary).
5. Verify replication functionality between the two servers.
6. Test replication on target objects.
7. Verify and checkpoint replication-ready environment.
8. Recycle Adaptive Server.

Run Regression Test Suites

Back-end regression test suite—production loads

1. Update back-end scripts.
2. Iteratively perform back-end regression testing.
3. Monitor and capture system dynamics (**sp_who, sp_lock, statistics io…**).
4. Verify and document SQL Server 10.x regression test.

Front-end simulation regression test suite

1. Iteratively do front-end simulation regression tests.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document SQL Server 10.x regression test.

Front-end Phase 1 and 2 regression test suite

1. Perform Phase 1 and 2 regression testing for local team.
2. Perform Phase 1 and 2 regression testing for users.
3. Monitor and capture dynamics (**sp_who, sp_lock...**).
4. Make “go / no go” call for the shadow SQL Server 10.x.

5. Verify and document SQL Server 10.x regression test.
6. Verify performance and functions on Server B's SQL Server 10.x.

Upgrade SQL Server on Server B (Shadow)

1. Alter *sybsystemprocs*.
2. Perform pre-upgrade verification.
3. Upgrade Server B to Adaptive Server 11.5.
4. Load Adaptive Server 11.5 environment from media dumps.
5. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**).
6. Set baseline Adaptive Server 11.5 configuration parameter values.
7. Perform Release 10.x dump from Server A.
8. Load Adaptive Server 11.5 with the Release 10.x dump.
9. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**) on Adaptive Server 11.5 databases. Be sure to verify the **dbcc** log and error log.
10. Recycle Adaptive Server.

Run Post-upgrade Regression Test Suites on Server B

Back-end regression test suite—production loads

1. Perform the back-end regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock, statistics io...**).
3. Verify and document this SQL Server 10.x regression test.

Front-end simulation regression test suite

1. Perform front-end simulation regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document this SQL Server 10.x regression test.
4. Load Adaptive Server 11.5 from media dump.

Front-End Phase 1 and 2 Regression Test Suite

1. Perform Phase 1 and 2 regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document this SQL Server 10.x regression test.

Other Testing

Verify 11.5 performance and functions.

Run User Acceptance Tests on Adaptive Server 11.5 (Server B)

1. Production testers perform user regression testing of shadow Release 11.5.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document this Adaptive Server 11.5 regression test.
4. Determine if Release 11.5 is “go / no go”.

Shift Production Users to Adaptive Server 11.5 (Server B)

1. Ensure that there is no production activity.
2. Perform a Release 10.x dump from Server A (production).
3. Load the dump onto Server B (shadow).
4. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**) on the Release 10.x databases just loaded to Server B. Be sure to verify the **dbcc** log and error log
5. Switch the IP address, and rename the machines and servers.
6. Run user testing and verification.

Perform Final Steps

1. Enable replication (Server B to Server A).
2. Start production users on Release 11.5 (Server B).

Cutover Migration Task List Example

This sample task list covers a cutover migration approach to upgrade SQL Server 10.x to Adaptive Server 11.5. The scenario for this task list is:

- The mid-sized company requires a simple, somewhat fault-tolerant upgrade.
- In the event of failure, the company depends on nightly backups. The system cannot be down more than one hour and cannot afford more than eight hours of data loss.
- The environment includes development, test, and production systems.

This example does not repeat the preparation steps given in previous examples.

Set Up Adaptive Server 11.5 Environment on Development System

1. Configure the physical drives and local array, duplicate the environment of the Release 10.x development system.
2. Perform a dump of the Release 10.x development system.
3. Install Adaptive Server 11.5.
4. Configure Adaptive Server 11.5 on the development system, duplicating the Release 10.x development environment.
5. Update the interfaces file.
6. Create the databases on Adaptive Server 11.5.
7. Load the dump of the Release 10.x development system.
8. Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on Adaptive Server 11.5 databases. Be sure to verify the **dbcc** log and error log.
9. Synchronize user IDs between the Release 10.x and 11.5 development systems.
10. Run **checkpoint** on the Release 11.5 environment.
11. Shift development to Adaptive Server 11.5. Developers begin verification and new feature development.

Define Test/Acceptance Criteria—Regression Test Suites

Front-end simulation regression test suite

1. Identify target user functions.
2. Capture and map SQL for target user functions.
3. Encapsulate SQL for user functions.
4. Create front-end simulated test suite (**showplan**, **stat io**, and **stat time** wrappers).

Front-end regression test suite

1. Identify front-end test scenarios.
2. Understand front-end application and acceptance / test procedures.
3. Document functional test approach.
4. Compose front-end test mix matrix.

Define Release 11.5 to 10.x Fallback Procedures on Test System

1. Create the fallback scripts to re-create the Release 10.x environment.
2. Document fallback procedures.

"Baseline" Release 10.x Environment on Test System

Recycle SQL Server 10.x.

Run Regression Test Suites on Release 10.x Test System

Front-end simulation regression test suite

1. Iteratively perform front-end simulation regression testing.
2. Monitor and capture system dynamics (**sp_who**, **sp_lock**...).
3. Verify and document this SQL Server 10.x regression test.

Front-end Regression Test Suite

1. Perform regression testing for local team.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Make the "go / no go" call for the Release 10.x test system.
4. Verify and document this SQL Server 10.x regression test.
5. Verify performance and functions on the test SQL Server 10.x.

Upgrade Release 10.x Test System to Release 11.5

1. Perform a dump of the Release 10.x databases.
2. Before the upgrade, run **dbcc** commands (**checktable, checkalloc, checkcatalog**).
3. Alter the *sybsystemprocs* database.
4. Perform a pre-upgrade verification.
5. Upgrade the test system to Release 11.5.
6. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**) on the Release 11.5 databases. Be sure to verify the **dbcc** log and error log.
7. "Baseline" the Release 11.5 configuration.

Run Regression Test Suites on Release 11.5 Test System

Recycle Adaptive Server before starting tests.

Back-End Regression Test Suite—Production Loads

1. Iteratively perform the back-end regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock, statistics io...**).
3. Verify and document Adaptive Server 11.5 regression test.

Front-End Simulation Regression Test Suite

1. Perform front-end simulation regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Make the "go / no go" call for the 11.5 system.
4. Verify and document this SQL Server 10.x regression test.

Front-End Regression Test Suite

1. Perform user regression testing.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document Adaptive Server 11.5 regression test.

Other Testing

Verify 11.5 performance and functions on the test system.

Run User/Acceptance Tests on the Release 11.5 Test System

1. Testers perform user regression testing of Release 11.5.
2. Monitor and capture system dynamics (**sp_who, sp_lock...**).
3. Verify and document the Release 11.5 regression test.
4. Make the Release 11.5 "go / no go" decision for the production system.

Execute Release 11.5 to 10.x Fallback Procedures on Test System

1. Shut down the Release 11.5 test system.
2. Re-create the Release 10.x test system.
3. Re-create the Release 10.x environment, using your re-creation scripts and procedures.
4. Load the Release 10.x production system dumps.
5. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**) on the Release 10.x databases. Be sure to verify the **dbcc** log and error log.
6. Run **checkpoint** on the Release 10.x test system.

Upgrade Production Server from Release 10.x to 11.5

1. Perform a dump of Release 10.x databases on the production system.
2. Before the upgrade, run **dbcc** commands (**checktable, checkalloc, checkcatalog**).
3. Alter the *sybsystemprocs* database.
4. Perform a pre-upgrade verification.

5. Upgrade the production system to Release 11.5.
6. Run **dbcc** commands (**checktable, checkalloc, checkcatalog**) on the Release 11.5 databases. Be sure to verify the **dbcc** log and error log.
7. "Baseline" the Release 11.5 configuration on the production system.
8. Perform user testing and verification.

Perform Final Steps

1. Start production users on the Release 11.5 production system.
2. Upgrade the test system to Release 11.5.

Staged Cutover Task Overview

This sample task overview covers a staged cutover migration approach to rebuild SQL Server. The scenario for this task list is:

- The large company has a mature client/server environment with multiple applications residing on a single SQL Server.
- In the event of failure, the company depends on nightly backups and transactions dumps. The system cannot be down more than one hour and cannot afford more than two hours of data loss.
- The environment includes development, test, and production platforms. It requires enough space for duplicate servers on each platform.
- The object-level rebuild strategy necessitates moving the data, which requires system downtime for the target application.

Tasks

The high-level tasks in this scenario are:

1. Configure Adaptive Server 11.5 on the development system, duplicating the SQL Server 10.x development configuration.
2. Migrate the development objects to the Release 11.5 development system.
3. Construct regression test suites.
4. Construct **bcp** scripts to move object deltas.
5. "Baseline" the Release 10.x test environment.

6. Configure a duplicate Adaptive Server 11.5 on the test platform.
7. Migrate the test / acceptance objects to the Release 11.5 test system.
8. Conduct regression test suites on the test system.
9. Verify synchronization of objects between the Release 10.x and 11.5 test systems.
10. Conduct user acceptance testing on the Release 11.5 test system.
11. Configure a duplicate Adaptive Server 11.5 on the production platform.
12. Migrate the production objects to the 11.5 production system.
13. Move the production users to the Release 11.5 production system.
14. Use your **bcp** scripts to resynchronize objects between the Release 11.5 and 10.x production systems.

E Migration Plan Checklist

The following checklist, produced by Sybase Professional Services, can be used to verify that you have covered all important issues in your migration plan. For more information on writing a migration plan, see Chapter 4, “Prepare: Writing a Plan and Getting Ready to Migrate”.

Logical Data Architecture

Does the logical data architecture include:

- A graphical representation of the data model from the logical architecture
- A usable map (like a CRUD matrix) of how data is used by the organization
- An approach for how data will be used by the organization
- An approach for how data will be distributed across platforms, locations
- An approach for how replicated data will be maintained
- An approach for how legacy data will be synchronized
- An approach for how data stores will be accessed, updated and purged

Logical Application Architecture

Does the logical application architecture include:

- A list and brief description of the required RPCs and stored procedures needed to support the new IT architecture
- List of any potentially sharable functions
- List of shared services such as initialization, termination, global edits, error handling, logging, monitoring
- An approach and a graphical representation of how application functionality will be split between servers, clients and middleware
- List of functional controls that need to be in place in the new architecture, for example audit procedures

- A description of how the new IT architecture applications will integrate with legacy applications
- Any graphical user interface standards that the new IT architecture applications must comply with
- A list of services that the new IT architecture applications must interface with, such as image, voice-mail, email, word processing, printing, fax or file transfer mechanisms
- An approach for how the performance expectations for the new IT architecture applications will be met
- An approach for how the availability requirements for the new IT architecture applications will be met

Logical Technology Architecture

Does the logical technology architecture include:

- Vendor-independent description (functions and features) and graphical representation of hardware and software components in the new systems infrastructure
- Information about expected network loads at normal and peak processing periods
- Information about any upgrades necessary to the existing network infrastructure, including information on specific carrier types
- Information on all connected nodes (workstations, database servers, gateways, etc.), including:
  - Projected quantities
  - Role in the new IT architecture (client, service provider etc.)
  - Platform characteristics for protocol handling, storage capacities, performance, fault tolerance and security
- Information on and graphical representation of the geographical location of the system's infrastructure components
- An approach for how the availability requirements will be met in terms of hardware / software capabilities, like fault tolerance, hot standbys, etc.

Logical Support Architecture

Does the logical support architecture include:

- Information on systems management procedures to be upgraded or put in place for:
  - Software distribution
  - Performance and fault monitoring
  - Fault management
  - Disaster recovery
  - Production approval and access control
- New support organization, including roles and responsibilities
- Staffing and training plans
- Strategy for meeting needs for support coverage (location and number of shifts)
- Strategy for meeting needs for required response time for problem resolution

Migration Strategy Design

Does the migration strategy include:

- An implementation sequence or a transition plan for candidate applications, that shows the evolution of the new IT architecture
- Information on key migration constraints, such as, data conversions and critical interfaces to legacy systems
- Information on bridging routines to keep legacy systems synchronized while the new IT architecture is evolving
- Information on methodologies, techniques and tools to be used in application development environments
- Initial strategy for putting new IT architecture applications into production
- Staffing, skill and training requirements to construct and test new IT architecture applications
- Preliminary project plan for migration

Index

D

E

F

H

I

L

M

N

O

P

R

S

T

W