SYBASE®

Migration Guide

Adaptive Server® Enterprise

Version 15.0

DOCUMENT ID: DC34982-01-1500-01

LAST REVISED: September 2006

Contents

About This Book

This manual describes how to migrate to Adaptive Server® Enterprise.

Audience

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

How to use this book

- Chapter 1, “Documenting Business Requirements,” helps you organize business information required for an effective migration plan.
- Chapter 2, “Documenting Your Environment,” provides guidelines for documenting system hardware and software in the Adaptive Server production environment.
- Chapter 3, “Writing a Migration Plan,” discusses migration methods and planning changes you may need to make to your system resources, applications and system administration procedures.
- Chapter 4, “Making Required Application Changes,” covers those issues that may affect the execution of applications or that might require coding changes.
- Chapter 5, “Making Database Administration Changes,” discusses changes to Adaptive Server system administration that may cause problems if you are not prepared for them.
- Chapter 6, “Ensuring Stability and Performance,” helps you to evaluate testing methods and develop a testing plan.
- Appendix A, “Worksheets for Your Current Environment,” provides guidelines and sample worksheets for gathering information needed to plan for migration.
- Appendix B, “Sample Migration Task Lists,” provides samples of completed task lists for general, parallel, and cutover migration. It includes a sample task list template and staged cutover task overview.
- Appendix C, “Migration Issue Checklists,” contains checklists that may be helpful as your write migration plans.

- Appendix D, “Pre-Upgrade Checklist,” contains a checklist for preparing for an upgrade in ascending order from earliest tasks to the last task that precedes the upgrade.

Related documents

The Adaptive Server Enterprise documentation set consists of the following:

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

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library.
- The *installation guide* for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What’s New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 15.0, the system changes added to support those features, and changes that may affect your existing applications.
- *ASE Replicator User’s Guide* – describes how to use the Adaptive Server Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.
- *Component Integration Services User’s Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- The *Configuration Guide* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.
- *Full-Text Search Specialty Data Store User’s Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server User’s Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, functions, and stored procedures in the Adaptive Server database.

- *Job Scheduler User's Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Messaging Service User's Guide* – describes how to use Real Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.
- *Monitor Client Library Programmer's Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server User's Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Guide* – is a series of four books that explains how to tune Adaptive Server for maximum performance:
  - *Basics* – the basics for understanding and investigating performance questions in Adaptive Server.
  - *Locking* – describes how the various locking schemas can be used for improving performance in Adaptive Server.
  - *Optimizer and Abstract Plans* – describes how the optimizer processes queries and how abstract plans can be used to change some of the optimizer plans.
  - *Monitoring and Analyzing* – explains how statistics are obtained and used for monitoring and optimizing performance.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book (regular size when viewed in PDF format).
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL information:
  - *Building Blocks* – Transact-SQL datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
  - *Commands* – Transact-SQL commands.
  - *Procedures* – Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.

- *Tables* – Transact-SQL system tables and dbcc tables.

- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Full-size available only in print version; a compact version is available in PDF format.
- *Transact-SQL User's Guide* – documents Transact-SQL™, the Sybase-enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the pubs2 and pubs3 sample databases.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – describes the Unified Agent, which provides runtime services to manage, monitor, and control distributed Sybase resources.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services User's Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.
- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that comprise XML Services.

Corresponding documentation to migration

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?* 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.

The following table gives general guidelines for relating migration phase to Sybase documentation:

Document	Read during	Range of tasks covered
• *What's New in Adaptive Server* • *Migrating to Adaptive Server Enterprise 15.0*	Planning/preparation prior to upgrade	Assessing current system Planning migration Making applications compatible Updating DBA procedures
• Release Bulletins	Planning/preparation prior to upgrade	Finding information needed to avoid upgrade problems, including problem reports, special installation issues, and compatibility issues
• Installation guide	Upgrade preparation and implementation	Preparing system to upgrade Installing software Performing upgrade tasks
• *System Administration Guide* • *The Performance and Tuning Guide*	Planning/preparation prior to migration and testing after upgrade	Planning system design for Adaptive Server 15.0 Monitoring and tuning system for increased performance

For additional information about Sybase database products, go to http://www.sybase.com/support/manuals.

Other sources of information

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

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

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

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

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

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

- Sybase Consulting offers services to customers planning to migrate to Adaptive Server 15.0. Sybase Consulting 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.
- Sybase offers a wide range of services to assist in the planning and execution of a migration to Adaptive Server 15.0. Using the migration methods developed and tested by Sybase can help reduce the time and risks associated with a migration.
- See the Sybase Education page for information about courses on Sybase products. For Adaptive Server release 15.0 courses, see the Database Servers at http://www.sybase.com/education/coursecatalog/databaseservers page.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ Finding the latest information on product certifications

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter select a product, platform, and timeframe and then click Go.
4. Click a Certification Report title to display the report.

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

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

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

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

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

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

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.
5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following sections describe conventions used in this manual.

SQL is a free-form language. There are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1 shows the conventions for syntax statements that appear in this manual:

Table 1: Font and syntax conventions for this manual

Element	**Example**
Command names, procedure names, utility names, and other keywords display in sans serif font.	select sp_configure
Database names and datatypes are in sans serif font.	master database
File names, variables, and path names are in italics.	*sql.ini* file *column_name* *$SYBASE/ASE* directory

Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italics in Courier font.	`select` *`column_name`*
	    `from` *`table_name`*
	    `where` *`search_conditions`*

Type parentheses as part of the command.	compute *row_aggregate* (*column_name*)
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates "is defined as".	`::=`
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	`[cash | check | credit]`
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	`cash, check, credit`
The pipe or vertical bar(|) means you may select only one of the options shown.	`cash | check | credit`

An ellipsis (...) means that you can *repeat* the last unit as many times as you like.	`buy thing = price [cash | check | credit]`
	`[, thing = price [cash | check | credit]]...`
	You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  ```
  sp_dropdevice [device_name]
  ```

  For a command with more options:

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

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

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

  ```
  select * from publishers
  ```

- Examples of output from the computer appear as follows:

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

(3 rows affected)
```

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

Adaptive Server sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. For more information, see the *System Administration Guide*.

Accessibility features

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

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

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

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

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.

CHAPTER 1

Documenting Business Requirements

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.

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

Creating an information-flow diagram

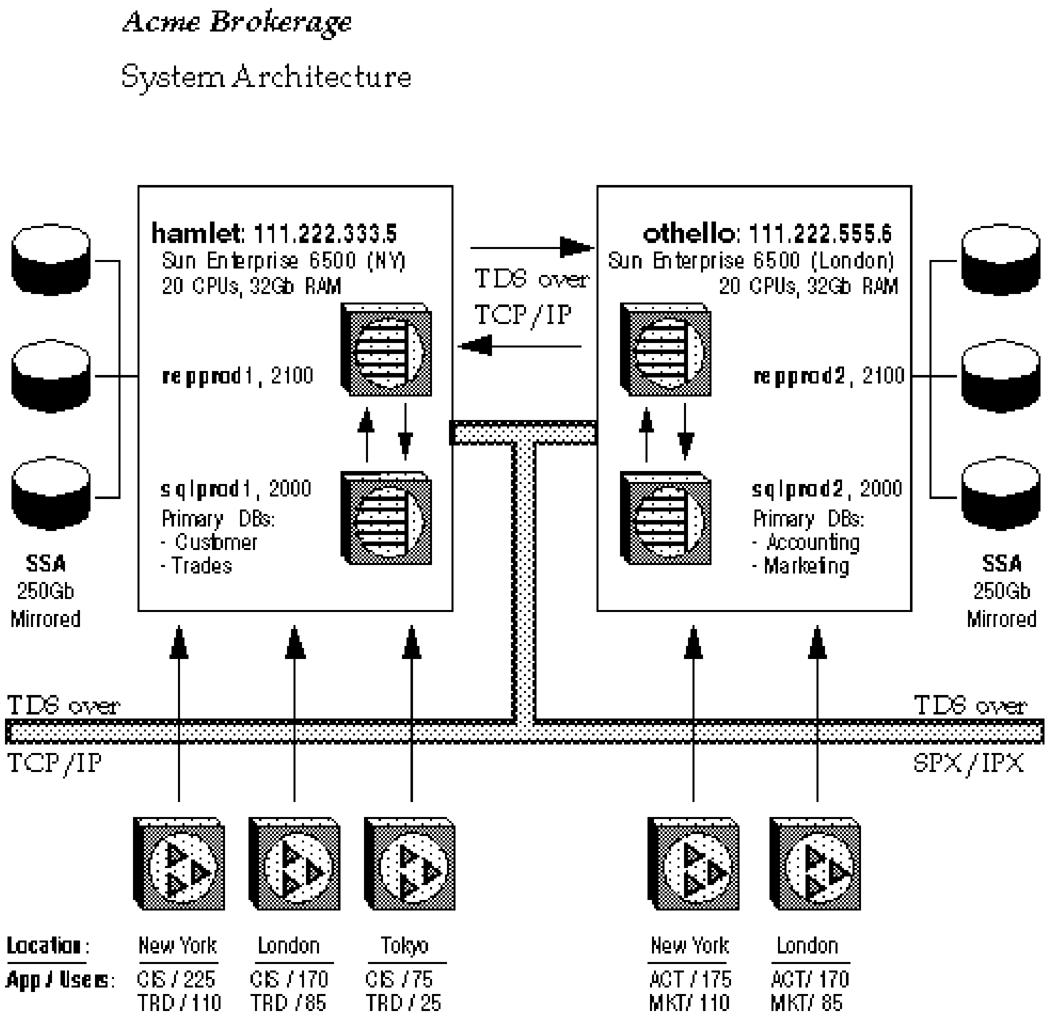
Create a diagram (or table or written description) that illustrates the flow of information on your system. This serves as a reference for the migration team. Include the following information:

- Servers, including file, print and application servers. For each server, include:
    - 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 Figure 1-1 which shows a high-level view of the Acme Brokerage, a firm with business in several cities and two main computing centers:

Figure 1-1: Diagram of brokerage information-flow



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

Identifying your operational business requirements

This section suggests ways to document your operational business requirements. These baseline requirements can help you plan the migration and develop success criteria.

- Availability requirements
- Database change metrics
- Database dump details
- Maintenance procedures
- Service-level requirements
- Transaction profile

Availability requirements

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

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 row counts 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:

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, using a table similar to this example:

Database name	Frequency of dbcc checkdb and dbcc checktable	Frequency of dbcc checkalloc and dbcc tablealloc	Frequency of update statistics	Frequency of dbcc checkstorage	Frequency of monitoring utilization
master	every night			every night	
TRD	every weekend	different tables “round-robin” every night	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	different tables “round-robin” every night	every hour
ACT	every weekend	different tables “round-robin” every night	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:

Application name	Type of application	App langs	Client machines	No. of con-current users	DBs accessed	Availability require-ments (per day)	Perform-ance (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 and record transaction profiles. Save showplan and dbcc output for critical transactions. You will use this information as a baseline for post-upgrade testing, as described in Chapter 6, "Ensuring Stability and Performance."

Create a document similar to the following:

App name	process (xact) name	Type of proces sing	Xact priority	Freq per user (per hour)	Source code	Required average response time	Required max 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	Embedd ed 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

---

Note If you are upgrading from Adaptive Server version 12.0 or later, you can save abstract query plans for your critical queries. For more information, see the *Performance and Tuning Guide*.

See the *Performance and Tuning Guide* for more information on gathering transaction statistics. Also see ASE Migration Resources Web page at http://sybase.com/support/techdocs/migration for TechNotes and white papers on query processing.

Recording current performance metrics

Record as much of the following performance information as possible. Sybase provides monitoring capabilities in the form of Monitor Server, monitoring tables, 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 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.

    - Use the monitoring tables to gather statistical and analytical information about the state of the 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.

Once you have completed the upgrade to Adaptive Server release 15.0 , use query metrics utility to compare prior abstract plans with current abstract plans.

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

Recording additional business requirements

Document any other important operational business requirements, such as:

- Priority list of applications to be migrated
- Constraints
    - Whether to avoid year/quarter end processing
    - An acceptable amount of down time
    - Whether the upgrade take place over a weekend
    - Available staff
    - Other available resources, for example, Replication Servers, machines, tools, funding
- Application and data server dependencies
    - Whether multiple applications use the same Adaptive Server
    - Whether all applications on a server be migrated
- Vendor issues, for example whether your third-party applications are certified to run against Adaptive Server version 15.0?

CHAPTER 2 Documenting Your Environment

This chapter provides guidelines for documenting system hardware and software in the Adaptive Server production environment. During the planning phase of your migration you can use this information to identify resource issues.

Topic	Page
Hardware configuration	9
Physical memory utilization	12
Software configuration	13
Sybase configuration	14
Adaptive Server objects	16
Adaptive Server performance	17

See Appendix A, "Worksheets for Your Current Environment" for worksheets like the ones used in the examples in this chapter.

Hardware configuration

Document your hardware environment:

- General server hardware
- CPU resources per machine
- Disk configuration
- Network configuration
- Tape configuration

General server hardware

For every server machine, list:

- 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
  - Vendor’s Web page

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 Adaptive Server
- Other CPU-intensive processes sharing those processors
- List of processes and threads bound to specific CPUs
- List of processes and 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

Controller number	**Make and model**	**Firmware revision**	**Months in service**	**Transfer rate (Kb/sec)**
0	Sun Fire V440	1.50	9	7500
1	Sun Fire V440	1.00	6	10000

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

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

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:

Physical device name	Make and model	Firmware revision	Months in service	Protocols supported	Network address	Transfer rate (Kb/sec)
c0t0d0	Sun Fire V480	1.5	9	TCP/IP SPX/IPX	121.222.233.11	7500
c0t0d1	Sun Fire V480	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:

Physical device name	Make and model	Firmware revision	Months in service	Controller number	Capacity (Mb_)	Transfer rate (Kb/sec)
/dev/rmt/0	Sun Fire E10K	2.15	9	2	2000	500
/dev/rmt/1	Sun Fire E10K	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:

Name	Runtime memory usage calculation
Operating system	OS-specific
Adaptive Server	The value of max memory, as listed in the configuration file (for Adaptive Server version 12.0, use total memory).
Backup Server	Add the value of max memory as specified by the -m parameter in the Backup Server configuration file.
Replication Server™	See “memory limit” parameter.

Name	Runtime memory usage calculation
Middleware (for example, Enterprise Connect Data Access)	Specific to the product—see product documentation.
Other applications {List:}	Specific to the application.
Total memory required	
Total memory installed	

Software configuration

Document your software environment as described in the following sections:

- Operating system
- Applications

Operating system

List the following operating system information:

- Operating system name
- Release level
- Patch level
- 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, recent 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
- Vendor's Web page

Applications

Make a list of applications to be migrated to Adaptive Server version 15.0. For each application, note:

- Information about data and usage
    - Distributed data.
    - Whether data is warehoused or used in transaction processing? If used in transaction processing, data must 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 (for example triggers and stored procedures)

Sybase configuration

Document your Sybase configuration as described in the following sections:

- General information
- Database devices
- Databases and segments
- Dump devices

General information

Record the following Sybase information:

- Adaptive Servers and their *$SYBASE* home directories (*%SYBASE%* for Windows)

- Components and release levels (including EBFs)
- Page size configuration of the server
- Names and location of scripts to rebuild database environment
- All server configuration values; you can find these in the *.cfg* file.

Database devices

Record database device information, as in the following example:

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 as in the following example:

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:

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

Adaptive Server objects

Document the objects in your current Adaptive Server.

Locate or create the scripts necessary to re-create:

- 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 to extract and load data with bcp. You can use these scripts to set up your test environment and build a new production system. They may be needed if you plan to maintain two server systems at different release levels.

If you do not have scripts, there are several ways you can reproduce scripts or access the information needed to reproduce configuration and objects:

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

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

  See the *System Administration Guide* for information on system tables and objects and “Segment Remapping with load database When Moving a Database” at http://www.sybase.com/detail?id=1324 for information on using system tables to reconstruct databases.

- Use system stored procedures – For information about current Adaptive Server configuration, use sp_configure with no argument. It lists all configuration parameters and their values.

  Use sp_helptext for the SQL commands in a stored procedure.

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

- Use Sybase tools – You can reverse-engineer server objects using Sybase tools such as Sybase Central™, PowerDesigner™ and WorkSpace, or other third-party tools

Adaptive Server performance

Use sp_sysmon to gather information about Adaptive Server current utilization during peak and idle times (typically, a two to five-minute sample is sufficient, but knowledge of your particular environment is necessary to determine both the quantity and frequency of the samples taken). See “Monitoring Performance with sp_sysmon” in the *Performance and Tuning Guide: Monitoring* for information on using sp_sysmon.

You can also use monitoring tables to evaluate overall server performance. See "Monitoring Tables" in the *Performance and Tuning Guide: Monitoring*.

You can use third-party tools such as Database Expert to benchmark and document query performance in your current Adaptive Server configuration. For more information, see "Analyzing Performance Changes during Adaptive Server Migrations and Upgrades with Sybase Database Expert" at http://www.sybase.com/detail?id=1028449.

Third party tools can also be used to obtain and analyze both query- and server-level performance data.

CHAPTER 3 Writing a Migration Plan

After you have collected data about your current system, Sybase suggest that you write a migration plan. This chapter discusses migration methods and planning. In addition to choosing a migration method, you may need to bring your system resources to the level required by Adaptive Server version 15.0 and make changes needed in applications and system administration procedures.

Before beginning the upgrade process, consider these issues:

- Before you run Adaptive Server, you must first implement your site licenses with the SySAM utility. See the Adaptive Server *Installation Guide* for your platform for pre-installation planning and SySAM installation information. See the *Users guide: Software Asset Management 2.0* for information about SySAM. You can also see http://www.sybase.com/sysam for more information about SySAM and license issues.
- For Adaptive Server installations 11.5.x and older, Sybase recommends that you first upgrade to Adaptive Server version 12.0, then upgrade to version 15.0.
- Before you upgrade to the latest emergency bug fix (EBF) release and migrate your server, review the *README* file included with the EBF for general information, loading instructions, and specific fixes.
- Some versions of Adaptive Server include system roles to control features. If you have modified the values of user or role IDs to resemble system accounts to control user IDs, login IDs, or roles, make sure these IDs do not conflict with users or roles added with the new version of Adaptive Server.

During a database upgrade, the upgrade process may add a new role to the database. If this role and ID pair already exists in the server, the new role cannot be added, causing the entire upgrade to fail. Before upgrading Adaptive Server, review sysusers and sysroles to make sure that any user or role IDs that you manually inserted are within the normal Sybase user range.

- If you are using dump and load to upgrade to Adaptive Server 15.0, after the upgrade is complete, you must copy the existing login IDs and role information from the earlier version to the upgraded server before you load the first database to be upgraded.
- You must repartitioned tables after you complete the upgrade. You can manually unpartition these tables before running the upgrade to save time and effort during the upgrade. This is particularly helpful for tables for which the partitioning scheme is changed. See "Partition changes" on page 50 for more information partition changes.

Note for the upgrade process

Migrating to a 64-bit operating system or a larger page size

On some platforms, a 64-bit version of Adaptive Server is available. For some platforms, this is the only version available.

Adaptive Server version 12.5 and later allows you to create a server that uses a logical page size larger than 2K. See *What's New in Adaptive Server?* for more information.

Table 3-1 describes the steps for migrating Adaptive Server in both cases.

Table 3-1: Migrating special cases

Type of migration	Methods/tools available
Pre-15.0 32-bit server to 15.0 64-bit server	You can use the sqlupgrade utility. **Note** You can use the sqlupgrade utility only when you are changing release levels. To move from a 32-bit to a 64-bit system, start the 64-bit server against the 32-bit server installation.
15.0 32-bit system to 15.0 64-bit system	Create a new 15.0 installation and use bcp or dump and load to move data into the new server. You can also manually replace the binary to change to a 64-bit system. See the installation guide for your platform for more information.
Pre-15.0 2K page server to 4K, 8K, or 16K server	Changing Adaptive Server's schema from one page size to another is a database migration rather than an upgrade. See Chapter 6, "Migration Utility" in the *Utility Guide* for information about migration.

Alternative to dump and load for Adaptive Server 12.5.1

If you are upgrading from Adaptive Server 12.5.1 or later and plan to build a new server instead of upgrading an existing server, the following steps are faster alternative to dumping and loading the database. This method is fastest if you are using storage area network (SAN) technology, even if you are migrating to new hardware at the same time. The server you are building must have access to (or copies of) the physical devices of the old server, and the databases must be aligned with the devices. If a device contains fragments of more than one database, you may have to move or upgrade all the databases at the same time.

1. Either quiesce the 12.5.x database to a manifest file during testing, or unmount the database from the server.
2. If the 15.0 Adaptive Server is on another host, unmount the disk devices from the current host and mount them to the new host machine or copy the devices using SAN utilities.
3. Copy the manifest file to the new Adaptive Server 15.0 location.
4. Mount the 12.5.x database to the Adaptive Server 15.0 server using the manifest file. You may need to rewrite the new device paths for the server location.

5 Issue the online database command. The database upgrade begins when you bring the database online.

Example

The following example assumes that you are upgrading the testdb database using a manifest file named *testdb_manifest.mfst*.

1 Quiesce the 12.5.x database:

```
quiesce database for_upgrd hold testdb
for external dump
to "/opt/sybase/testdb_manifest.mfst"
with override
```

2 Copy the devices using disk copy commands (for example, dd), SAN utilities, or standard file system command.

3 When the device copy is finished, release the quiesce:

```
quiesce database for_upgrd release
```

4 If this is the final upgrade for this server (as opposed to a server for testing), shut down the server to prevent further changes.

5 Move the device copies to the new host machine and mount them as appropriate.

6 In the 15.0 server, issue the following to list the physical-to-logical device mappings:

```
mount database all from "/opt/sybase/testdb_manifest.mfst" with listonly
```

7 Use the results from step 6 to determine the new physical device mappings that correspond to the logical devices and mount the database in the 15.0 Adaptive Server:

```
mount database all from "/opt/sybase/testdb_manifest.mfst" using
"/opt/sybase/syb15/data/GALAXY/test_data.dat" = "test_data",
"/opt/sybase/syb15/data/GALAXY/test_log.dat" = "test_log"
```

8 Bring the database online and start the upgrade process:

```
online database testdb
```

Determining a migration approach

The best migration strategy for you depends on such factors as the cost of the effort, the type of business you do, the size of your databases, and available resources.

Table 3-1 highlights the advantages and disadvantages of each migration approach:

Table 3-2: Advantages and disadvantages for migration approaches

Approach	Advantages	Disadvantages	When used
Parallel with replication	Easy fallback to earlier version, you need not rebuild previous release databases. Minimal system down time.	Can be complex in OLTP environments. Replication Server must be set up, requiring extra hardware and software.	This approach is best for large 24/7 production databases maintaining high availability, when: • Rebuilding a release database may 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 good for resource-constrained environments. It is appropriate for large organizations only if you are able to schedule sufficient down time, for instance, a long weekend.
Phased cutover	Low risk with low development overhead. Especially conducive to testing.	May require additional resources—either more memory or a second 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 these sections:

- Parallel with replication
- Cutover without replication

- Phased cutover

Note This guide does not discuss other parallel migration approaches; such as running two systems in parallel where you must maintain both 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 to verion 15.0 first. Upgrade the production system after testing. See Chapter 6, "Ensuring Stability and Performance" for more information about testing.

Parallel with replication

Method

The following general method can be used to perform a parallel upgrade with replication:

1. Install a new copy of Adaptive Server 15.0.
2. Copy in or load pre-release 15.0 databases.
3. Use Replication Server to maintain both sets of databases. The 15.0 system becomes the primary server, and you maintain the pre-15.0 system as a warm standby.

Fallback

Plan for all users to reconnect to the earlier version server after you take Adaptive Server verion 15.0 off line. Make TCP/IP address and port changes where appropriate.

Test the fallback process as part of the application test suite (these are a series of tasks that test your application). This suite should do both of the following:

- Insert data into Adaptive Server version 15.0. The data must be replicated and available in the earlier server.
- Execute the fallback script.

  Consider making daily bcp dumps of the databases. To fall back, load the dumps into the earlier server. Keep in mind:

  - You may need to modify the databases to support incremental bcp dumps.
  - The earlier server cannot read version 15.0 backup files. Create bcp or other scripts to move tables back to pre-15.0.

Do not apply schema enhancements.

For information about scheduling backups of user databases, see the *System Administration Guide*.

Additional tips:

- 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 client fallback time in the calculation.
- For sites that are perpetually running, load time delays can impact synchronization. Consider replicating to Adaptive Server 15.0, then switching servers.

Application test suite

Execute this suite *before* switching users to the new system.

Since the parallel-with-replication approach is best for high availability applications, it is imperative that the test suite address both update correctness and performance acceptability.

See Chapter 6, “Ensuring Stability and Performance” for more information on testing.

Note After successful validation, consider having users enter production queries. You should perform your tests with a realistic production load. A good time to perform the tests 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 are to have issues during migration.

To ensure correct updates and acceptable performance, test the replicated environment.

Environment

The environment used for Adaptive Server vesion 15.0 must be more powerful than for earlier version to handle query and replication loads. See the Replication Server *Configuration Guide* for your platform for more information.

Be sure to account for any increased release 15.0 memory requirements that apply to your configuration. For more information, see:

- The installation guide for your platform for basic RAM requirements.
- Chapter 5, “Making Database Administration Changes.”.

- The Adaptive Server *System Administration Guide* for information about configuring memory and data caches.
- The *Performance and Tuning Guide* for Information on how to configure memory for performance.

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 15.0 issues.

For a production system, be prepared to postpone or fall back, if needed.

Cutover without replication

Method

Upgrade all databases to release 15.0 at the same time. A cutover without replication is common in small organizations for development or production servers. It may be used in larger organizations when you can schedule sufficient downtime.

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 client fallback time 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 fall back, 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 6, "Ensuring Stability and Performance" for more information on testing.

Note You might want to validate over a long weekend, if possible.

Fallback after cutover

Consider making daily bcp dumps of the databases. You can then load the dumps into the earlier server to fall back. Keep in mind:

- You may need to modify the databases to support incremental bcp dumps.
- Earlier servers cannot read release 15.0 backup files. You must create bcp or other scripts to move tables back to pre-release 15.0.
- Do not apply Adaptive Server 15.0 schema enhancements.

  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 15.0 memory requirements that apply to your configuration. For more information, see:

- The installation guide for your platform for basic RAM requirements.
- Chapter 5, "Making Database Administration Changes."
- Details on configuring memory and data caches in the Adaptive Server *System Administration Guide*.
- The *Performance and Tuning Guide* for information on how to configure memory for performance.

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 15.0 issues.

For a production system, be prepared to postpone or fall back, if needed.

Phased cutover

Method

Change only one application and database to release 15.0 at a time.

Fallback

Consider making daily bcp dumps of the databases. To fall back, you can then load the dumps into the earlier server. Keep in mind:

- You may need to modify the databases to support incremental bcp dumps.
- The earlier server cannot read release 15.0 backup files. You need to create bcp or other scripts to move tables back to the earlier server.
- Do not use release 15.0 schema enhancements 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 and libraries for both releases.
- Make sure the applications are using the correct server.
- After successful validation, consider having users enter production. A good time to do so is after hours or during production lulls.

  See Chapter 6, “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.

The earlier server cannot read release 15.0 backup files. You must create bcp or other scripts to move tables back to pre-release 15.0.

Note Do not use release 15.0 schema enhancements until the conversion succeeds.

Environment

Be sure to account for any increased release 15.0 memory requirements that apply to your configuration. For more information, see:

- The installation guide for your platform for basic RAM requirements.
- Chapter 5, “Making Database Administration Changes.”
- Details on configuring memory and data caches in the *System Administration Guide*
- The *Performance and Tuning Guide* for information on how to configure memory for performance.

  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 15.0 issues.

For a production system, be prepared to postpone or fall back if needed.

Be sure to notify users when you will be taking applications or databases offline.

Writing a migration plan

Produce a project plan that documents:

- Migration strategy – the most appropriate method for your site.
- Fallback – what to do if migration fails. The plan you evolve will be site-specific, but some general issues are discussed in Chapter 6, “Ensuring Stability and Performance”.
- Application test suite – what validation and performance testing to perform for acceptance. See Chapter 5, “Making Database Administration Changes.” for guidance.
- Bridging – ways to minimize the impact to users during the migration. Refer to the business requirements you gathered in Chapter 2, “Documenting Your Environment.”
- Environment – additional resources and changes to the environment needed based on the information you gathered in Chapter 2, “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 2, “Documenting Your Environment.”

You may also want to include the following as part of your migration plan:

- A work breakdown that lists tasks chronologically and assigns them to specific roles like the one in Appendix B, “Sample Migration Task Lists.”
- Specification for application changes. The details of needed application changes are discussed in Chapter 4, “Making Required Application Changes.”

Building the Adaptive Server environment

After deciding the best migration approach for your system, begin preparing your environment for Adaptive Server release 15.0:

- Update hardware resources
- Verify operating system version and EBF level
- Review Adaptive Server interoperability with other Sybase products
- Use Sybase Product Download Center
- Implement licensing environment
- Update applications and system administration procedures
- Create Migration Scripts
- Create a test environment. See Chapter 6, “Ensuring Stability and Performance” for more information on the test environment

Update hardware resources

Evaluate hardware resource needs according to your chosen method of migration. For example, if you plan to use the parallel-with-replication method. you may need additional disk space to install a secondary system, a Replication Server for the high availability environment, a disk farm for fallback strategies, and additional memory.

The release bulletin for your platform contains information about any hardware requirements.

See the physical memory requirements noted in the installation guide for your platform. See Chapter 5, “Making Database Administration Changes,” for more information on memory requirements.

See the Adaptive Server *System Administration Guide* for information on memory and backups.

Verify operating system version and EBF level

Ensure that the operating system is at the proper version and level to run Adaptive Server release 15.0. Be sure that you have installed the most recent operating system patches to ensure that you have the latest bug fixes.

See the Adaptive Server Migration Resources Web page at at http://sybase.com/support/techdocs/migration for updates about recommended Sybase EBFs (bug fixes).

Note If you need to perform an operating system upgrade, do so before migrating. To avoid introducing unrelated errors into the migration process, test the new system to make sure it is working properly..

Review Adaptive Server interoperability with other Sybase products

To ensure that the versions of other Sybase products at your site are compatible with Adaptive Server 15.0, see the product and platform interoperability section of the release bulletin for your platform.

Because of system changes in Adaptive Server 15.0 (particularly in the query plan output), you must use DBExpert release 15.0 to migrate to Adaptive Server 15.0.

Use Sybase Product Download Center

Customers typically installed earlier releases of Sybase software using CDs. However, with the widespread use of electronic software distribution, most customers download Adaptive Server 15.0 from the Sybase Product Download Center (SPDC). The downloaded software is in the format of a CD image, so you can burn it to a CD and install it from this image, if necessary. To avoid installation problems, consider the following:

- Make sure you use a short path to a download location (for example, */sybase/downloads* or *c:\sybase\downloads*). The path should not contain any special characters, such as the space character. On Windows operating systems, do not use directories such as *C:\Program Files\Sybase\downloads* (the space between “Program” and “Files” is a problem). Using bad paths or using the wrong version of the JRE results in “class not found” installation failure errors.

- Adaptive Server ships with the correct JRE for the InstallShield utility. If you have other Java environments installed (for example, JDK or JRE), and have environment variables referencing these locations (for example, *JAVA_HOME*, or if you include *java* in your *path* environment variable for frequent compilations), disable these environment variables in the shell from which you start the installation.
- Use the GNU gnuzip utility to uncompress the CD images on UNIX systems. These images have been compressed using the GNU zip utility, and decompressing them with standard UNIX compression utilities can corrupt the images.
- Use GNU tar utility to extract the archive files for the CD image. The standard UNIX tar utility may not extract the files correctly.
- Many of the hardware vendors support a variety of mount options for their CD drives, see the installation guide to make sure you are using the correct mount options specified for your platform.

Implement licensing environment

Even before the government implemented the Sarbanes-Oxley regulations, many customers requested that Sybase implement strong license management and reporting capabilities. Sarbanes-Oxley made this even more important because it holds CEOs of corporations responsible for any financial wrongdoing, including misuse of software assets. Because the banking industry is one of the most heavily watched industries, and given Sybase’s position within the financial community, Sybase adopted a more stringent licensing management implementation.

Since Sarbanes-Oxley applies to all publicly traded companies, Sybase opted to design SySAM version 2.0 so that licensing compliance is easier to measure for corporate users. One of the critical components of SySAM 2.0 is a reporting feature that allows information technology managers to monitor and report on license compliance to corporate officers. At no time does the SySAM software report license usage to Sybase. Any long-term use of Sybase software beyond the licensing agreements must be reconciled with a Sybase sales representative.

For a complete list of the SySAM features, see the *User’s Guide Sybase Software Asset Management* and the configuration guide for your platform.

How many license servers?

The number of license servers you deploy is a function of how you wish to handle the geographic distribution of your hardware or the divisional separation of your project or business unit. Organizations that are widely dispersed typically have at least one license server per geographic region. While SySAM can support more than one license server, it is not technically necessary because the load on a license server is extremely light.

How many servers per license?

Although SySAM allows you to create redundant license server nodes to ensure they are always available, this is necessary only if you expect that a single license server will be unavailable for an extended period of time (for example, the grace period of 30 days). For Adaptive Server 12.5, customers licensed each server individually. In Adaptive Server 15.0, if you are using the served license (that is, licenses that are served from a central location, also known as network licensing) model, the licenses are managed centrally. In the served license model, you register only the license server node with Sybase, not the individual servers. The requirement to register the license server node ID with Sybase is not for reporting requirements, but because the license server node ID is used to generate the license key.

What if I deploy my own image?

Some customers prefer to build their own deployment image for Sybase installations, using internally certified Adaptive Server versions and patch levels, containing the software suite in “tarballs” or other electronic packaging. If you use this system to implement multiple license servers, the only change to your deployment images is that, after the image is deployed, you must specify which license server Adaptive Server needs to contact for licensing. You can include this information in each deployment image for global installations since different builds probably have different localization requirements. However, you must change the license server *host_name* in the first line of the deployed license file, similar to this:

```
SERVER my_license_srvr 00096b138c70
VENDOR SYBASE
USE_SERVER
```

What if I use remote servers?

If you use remote servers (particularly if they are outside a firewall), you can deploy license servers remotely, but this provides no benefit and is likely impractical. Instead, use the unserved license model in which license keys are read from a local file. For unserved licenses, the individual *hostID*s must be registered with Sybase. OEM Adaptive Server deployments use a separate licensing implementation for embedded systems, and Sybase does not expect that using unserved licenses is a frequent requirement for these systems.

What about separately licensed options?

In earlier versions of Adaptive Server, you licensed the Java, XML, and XFS (external file system or content management) options. Adaptive Server release 15.0 includes licenses for these options. Other options such as high availability, DTM, and others, require separate licensing. When you start Adaptive Server 15.0, it automatically checks for some of these included licenses.

What's included with the Developer's Edition?

The Adaptive Server Developer's Edition includes all non-royalty options from Sybase. Royalty options include the Enhanced Full Text Search, Real Time Data Services, and most of the tools such as DBXray and Sybase Database Expert. When you install the Developer's Edition and are asked to select the license model, select the unserved license model because the Developer's Edition uses the file-based licensing mechanism. Because the Developer's Edition keys are included in the product CD image, no further SySAM actions are required after installing the software.

Where do I go for more information?

Further information or assistance on SySAM is available at http://www.sybase.com/sysam. In addition, you can call Sybase Customer Service or Technical Support at 1-800-8SYBASE, or talk to your local Sybase Sales Team.

Update applications and system administration procedures

The following chapters in this manual describe Adaptive Server 15.0 upgrade issues:

- Chapter 4, "Making Required Application Changes"
- Chapter 5, "Making Database Administration Changes"

Migrating to Adaptive Server 15.0 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; choose a method to do this that suits your needs and resources. Possible choices include:

- Writing your own scripts to examine and change applications.
- Running the reserved word check for the 15.0 server and using some of the tools available such as the stored procedures sp_checkreswords and sp_procqmode. See the *Reference Manual: Procedures* for information about these stored procedures.
- Using the preupgrade utility to check for common upgrade problems. See the *Utility Guide* for more information.

Create Migration Scripts

Using the scripts you located, wrote, or reverse-engineered in Chapter 1, “Documenting Business Requirements”, write or edit the scripts to create your 15.0 Adaptive Server installation. You need:

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

CHAPTER 4 Making Required Application Changes

This chapter and Chapter 5, "Making Database Administration Changes," divide technical issues into those relevant to application developers and those relevant to database administrators. However, many issues are not exclusive to either of these roles and Sybase recommends that you read both chapters, whatever your role. This chapter covers those issues that may affect the execution of applications or that might require coding changes.

This chapter covers only those new features and changes that may cause unexpected behavior for system administrators. There are many more changes that have occurred in Adaptive Server over the past few releases. For a comprehensive listing of changes and new features, see *What's New in Adaptive Server*.

Note Testing and changing existing applications and system administration procedures are the most time-consuming part of your migration preparation. 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. For instance, you may want to develop your own plans and scripts, or you may prefer to contact Sybase for help.

The Adaptive Server Migration Resources Web page at at http://sybase.com/support/techdocs/migration contains migration TechNotes and white papers. Sybase Technical Support can help with technical problems such as bugs and error conditions.

Begin reading at the earliest section that includes information about your current version of Adaptive Server:

- Upgrading Adaptive Server release 11.5
- Upgrading from version 11.9.x
- Upgrading from release 12.0
- If you are upgrading from release 12.5

Reserved words

Reserved words can be used only by Adaptive Server, not by applications connecting to Adaptive Server. Most new versions of Adaptive Server contain new reserved words as a result of new object and commands.

Before you upgrade Adaptive Server, you must change all object names that contain reserved words. You must also change those names in your procedures, Transact- SQL scripts, and applications before you can run them against the upgraded server. To check the object names, use the reserved word check in sqlupgrade, which you can run without starting the upgrade. Alternatively, you can apply the *installupgrade* script from the *$SYBASE/ASE-15_0/scripts* directory against your older Adaptive Server and run the sp_checkreswords stored procedure on this Adaptive Server.

Note You cannot use the Sybase procedures for detecting reserved words in objects to find them in scripts and applications. Check scripts and applications separately.

You can use sp_rename to change names that contain reserved words can be changed with sp_rename, or you can enclosed them in double quotes. See the discussion on reserved words in the installation guide for your platform.

See the *Reference Manual: Blocks* for a complete list of reserved words. See *What's New in Adaptive Server Enterprise* for a list of reserved words added for each release.

Upgrading Adaptive Server release 11.5

You can upgrade to Adaptive Server release 15.0 only from Adaptive Server releases 11.9.x, 12.0.x, or 12.5.x.

If you are currently running Adaptive Server release 11.5, Sybase recommends that you upgrade first to release 12.0, then upgrade to 15.0.

Upgrading from version 11.9.x

This section discusses:

- ANSI joins
- Query processing changes

ANSI joins

Adaptive Server releases 12.0 and later support ANSI joins. Sybase recommends that you rewrite applications to use ANSI outer joins because they specify unambiguously whether the on or where clause contains the predicate. Transact-SQL syntax in previous releases was ambiguous in some cases.

ANSI syntax allows you to write either of these types of joins:

- Inner joins – the joined table includes only the rows of the inner and outer tables that meet the conditions of the on clause. The result set of a query that includes an inner join does not include any null supplied rows for the rows of the outer table that do not meet the conditions of the on clause. The syntax for an ANSI inner join is:

  ```
  select select_list
  from table1 inner join table2
  on join_condition
  ```

  For example:

  ```
  select au_id, titles.title_id, title, price
  from titleauthor inner join titles
  on price > 15
  ```

- Outer joins – the joined table includes all the rows from the outer table whether or not they meet the conditions of the on clause. If a row does not meet the conditions of the on clause, values from the inner table are stored in the joined table as null values. The where clause of an ANSI outer join restricts the rows that are included in the query result. ANSI syntax also allows you to write nested outer joins. The syntax for an ANSI outer join is:

  ```
  select select_list
  from table1 {left | right} [outer] join table2
  on predicate [join restriction]
  ```

  For example:

  ```
  select au_fname, au_lname, pub_name
  from authors left join publishers
  on authors.city = publishers.city
  ```

Query processing changes

The query processing and optimizer changes introduced in Adaptive Server release 12.0 are less extensive than those in 11.9.2, and should provide improved performance for most sites. These changes include:

- Predicate transformation and factoring
- Support for as many as 50 tables in a query
- Abstract query plans
- Increased optimization time
- like optimization enhancements
- Concrete identification

Some of the changes that occurred in the 12.0 release of Adaptive Server are superseded by changes that occurred in the 15.0 release of Adaptive Server. These changes are described in *What's New in Adaptive Server*.

For information on optimizer changes over recent Adaptive Server releases, see "An Introduction to Sybase Adaptive Server Enterprise's Modern Optimizer" at at http://my.sybase.com. This downloadable collection of optimizer-related articles published in the ISUG Technical Journal requires Acrobat Reader.

Predicate transformation and factoring

Predicate factoring provides significant performance improvement in queries with limited access paths, which include queries with few possible search arguments, joins, or or clauses that can be used to qualify rows in a table.

Additional optimization is achieved by generating new search paths based on join conditions, search clauses, and optimizable or clauses.

Predicate factoring helps query optimization by extracting optimizable clauses from predicates linked with or (difficult to optimize) and replacing them with and clauses (easier to optimize), thus providing the optimizer with more usable search arguments. The more search arguments available, the more information the optimizer has, and the more likely it is to chose an efficient plan.

Full Cartesian joins are avoided for some complex queries.

Example:

```
select * from lineitem, part
where ((p_partkey = l_partkey and  l_quantity >= 10)
or (p_partkey = l_partkey and l_quantity <= 20) )
```

becomes

```
select * from lineitem, part
where ((p_partkey = l_partkey and  l_quantity >= 10)
or (p_partkey = l_partkey and l_quantity <= 20) )
and  (p_partkey = l_partkey)
and  (l_quantity >= 10  or   l_quantity <= 20)
```

The addition of the conjuncts adds usable search arguments for the optimizer. If new conjuncts (and clauses) added by predicate transformation and factoring are found to be not useful as part of index access strategy (that is, for filtering) they are not used.

Query semantics do not change; the result set is the same. Predicate factoring cannot be turned off, and is completely transparent to the user. It is implemented as a new compiler phase, just before the start of the optimizer.

When it is implemented, trace flag 302 output shows additional costing blocks and showplan shows additional “Keys are” messages.

Support for as many as 50 tables in a query

Adaptive Server release 12.5.1 and later supports as many as 50 user tables and 14 worktables, increased from 16 user tables and 12 worktables in 11.9.2. However, the limits on number of referential integrity checks remain at 192 and the number of subqueries allowed remains at 16.

Note Adaptive Server release 15.0 supports 46 worktables in a query.

Abstract query plans

Abstract plans were introduced in Adaptive Server release 12.0 as a means for System Administrators and performance tuners to protect the overall performance of a server from changes to query plans, such as:

- Adaptive Server software upgrades that affect optimizer choices
- New Adaptive Server features that change query plans
- Changing tuning options such as the parallel degree, table partitioning, or indexing

Adaptive Server can capture query text and save an abstract plan for a query in a system table called sysqueryplans. Using a rapid hashing method, incoming SQL queries can be compared to stored query text, and if a match is found, the saved abstract plan is used to execute the query.

For more information on abstract query plans, see the *Performance and Tuning Guide: Optimizer and Abstract Plans*.

For a quick overview of how to use abstract query plans to troubleshoot performance problems after an upgrade, see the TechNote "Using and Maintaining Abstract Query Plans" at at http://my.sybase.com.

Increased optimization time

Queries with long chains of join keys may require additional time to optimize with Adaptive Server.

If the time required to optimize such a query is unacceptable, consider using an abstract query plan for the query.

Note The performance of these queries is improved with Adaptive Server version 15.0.

like optimization enhancements

Version 12.0 changed the costing for like clauses not migrated into search arguments, using a technique of generating more accurate cost estimates for queries that include a leading wildcard in a like clause.

This provided better selectivity estimates, resulting in better query plans. Previously, a like clause with a leading wildcard was estimated to qualify all rows in the column (a selectivity of 1.0) since there was no way to search for a match. This was not the most accurate way to cost such a clause.

Example, in this query in a version 12.0 server:

```
select ... from part, partsupp, lineitem
where l_partkey = p_partkey
and l_partkey = ps_partkey
and p_title like '%Topographic%'
```

The like string is compared with histogram cell boundaries.

A match occurs when the like term is found in a cell boundary by means of a pattern match search.

If no pattern match is found, selectivity is assumed to be 1 divided by the number of steps in the histogram. If the default of 20 cells is used, selectivity is 0.05 if there is now pattern match in the histogram boundary values.

If a pattern match is found in the cell boundaries, the selectivity is estimated to be the sum of the weights of all cells with a pattern match.

In either case, the resulting selectivity estimate is more accurate than in versions earilier than 12.0.

This also applies to queries with like clauses of the type `like “_abc”`, or `like “[ ]abc”`.

Concrete identification

Concrete identification enables Adaptive Server to verify chains of ownership between procedures, views, and triggers and the objects they reference in other databases. When a user creates an object, Adaptive Server associates both the owner’s database user ID (UID) and the object creator’s login name with the object in sysobjects. This information concretely identifies the object as belonging to that user, which allows Adaptive Server to recognize when permissions on the object can be granted implicitly. This may affect older applications that have cross-database queries, giving you permission errors.

See Chapter 17, “Managing User Permissions” in *System Administration Guide: Volume One* for more information about concrete identification.

Upgrading from release 12.0

This section discusses the changes for version 12.0 Adaptive Server that affect migration, including:

- Changes to Transact-SQL
- enable xact coordination configuration parameter
- Unlimited expressions in a select statement
- Wide columns and data truncation

Changes to Transact-SQL

In general, changes to existing Transact-SQL commands add options and do not create problems for applications when these changed commands appear. For a list of changed Transact-SQL commands, see *What's New in Adaptive Server Enterprise*.

enable xact coordination configuration parameter

Adaptive Server release 12.0 introduced the enable xact coordination configuration parameter, and is enabled by default in Adaptive Server version 12.5.

See Chapter 5, "Setting Configuration Parameters" in the *System Administration Guide: Volume One* for more information about enable xact coordination.

Unlimited expressions in a *select* statement

Adaptive Server version 12.5 has no explicit limit on the number of expressions in a select statement. The number is limited only by the available system memory

Wide columns and data truncation

Adaptive Server 12.5 introduced larger page and column sizes. This may produce unexpected results in existing applications where data truncation previously occured. In addition, if you plan to use larger data sizes, consider the ability of client software to handle the new sizes. This section is overview:

- Change in truncation behavior
- Wide columns and optimizer statistics
- Wide columns with col_length() and datalength()

Change in truncation behavior

Versions of Adaptive Server earlier than 12.5 had a column-length limit of 255 bytes. Adaptive Server allows you to create columns using char, varchar, binary, and varbinary data up to 16294 bytes long, depending on the logical page size your server uses. Because of this, data that was truncated at 255 characters for earlier versions is no longer truncated. If your application depended on this truncation, the result set it receives may no longer be accurate. In the following example, col1 and col2 are each 200 characters long, and col3 is 255 characters long:

```
select * from t1 where col1 + col2 = col3
```

If col1 and col2 each hold 200 characters, their concatenation results in a 400-character string. Earlier versions of Adaptive Server truncate this to 255 characters, and the sum of col1 and col2 might match col3. However, for the 12.5 release of Adaptive Server, the product of col1 and col2 is 400 characters, and will never match a col3 that is 255 characters long.

Character and binary expressions can produce a result up to 16384 bytes long. Data that is longer than this is truncated.

Wide columns and optimizer statistics

Adaptive Server uses only the first 255 bytes of data when placing statistics on a wide column. For example, only the first 255 bytes contribute to the column histograms in sysstatistics if you place an index on a column created as `char(500)`. For unichar columns, the first 127 chars are used to gather column distributions.

Wide columns with col_length() and datalength()

The commands col_length() and datalength() are built-in Transact-SQL functions that return database information. These functions can now return values greater than 225. If you use one of these functions in a query and assign the result to a variable, make the variable large enough to hold the value.

If you are upgrading from release 12.5

This section covers the following topics:

- Date and time datatypes

- Query and optimizer changes
- Unsupported trace flags
- Partition changes
- Changes for computed columns and function-based indexes
- Long identifier changes
- Error message changes

Date and time datatypes

In versions earlier than 12.5.1, Adaptive Server included only the datetime and smalldatetime date and time datatypes. Adaptive Server 12.5.1 added the date and time datatypes. If your application accesses tables using these datatypes, you must account for these datatypes in your scripts and applications.

Query and optimizer changes

Adaptive Server 15.0 has a new optimizer. Most queries run on this optimizer should run the same or better with few or no changes. However, test all applications, before you use the server in production, for the following issues:

- Because of changes in the parser, some queries may return a general syntax error (message 102) instead of `Syntax error at line #` (message 156).
- The maximum number of worktables per query increased from 14 to 46.
- Adaptive Server 15.0 orders a result set differently from earlier versions unless there is an explicit order by clause in the query:
  - Earlier releases returned result sets in sorted order when you used a group by clause, even without an order by clause. The grouping algorithm created a worktable with a clustered index, and grouping is based on inserting into this work table.
  - In Adaptive Server 15.0, the grouping algorithm uses a hash-based strategy, which does not generate a sorted result set. To generate a sorted result set, change queries to include an order by clause. This change complies with ANSI SQL standards.

    - Adaptive Server 15.0 includes alternative algorithms for relational operations: hash- or merge-based union, hash- or sort-based distinct, and so on. The execution plan determines the ordering of the result set of a query without an order by.

- Adaptive Server 15.0 increases the amount of time for query compilation because the query processing engine looks for more ways to optimize the query. However, there is a new timeout mechanism in the search engine that can reduce the optimization time if there is a cheaper plan.
- Adaptive Server 15.0 deprecates the enable sort-merge join and JTC configuration option, so you may see the query engine using merge joins unexpectedly. You can force the query engine to use the behavior from earlier versions, but the new in-memory sort operations improve the sort merge join performance. The query optimizer does not use merge joins if they are inappropriate. You can disable sort merge join at the session level with the set merge_join 0 command, or disable it with the allrows_oltp optimization goal.
- Adaptive Server 15.0 allows you to disable reformatting at the session level with the set store_index 0 command.

  See Chapter 17, "Managing User Permissions" in *System Administratin Guide, Volume One* for more information about using login triggers in Adaptive Server to modify the session level settings without modifying applications.
- Adaptive Server 15.0 improves algorithms that organize joins between compatible, but different, data types. You can use indexes even if the SARGs use different datatypes. However, do not use trace flag 291, which gives spurious erroneous result sets.
- Adaptive Server 15.0 does not support set statistics io for parallel queries, but is supported for nonparallel queries.

  You can get an accurate interpretation of the statistics by setting set statistics plancost on, which shows the parallel access nodes in the Lava operator tree.

See *Query Processing in Adaptive Server* for details about query processing in Adaptive Server 15.0.

Unsupported trace flags

Trace flags influence Adaptive Server behavior or correct issues. However, some trace flags display unexpected behavior, and all trace flags require caution. Many trace flags are specific to Adaptive Server release 12.5.x and are not necessary for Adaptive Server 15.0. Table 4-1 describes some of the commonly used trace flags and their relevance to Adaptive Server 15.0. Before you discontinue using a trace flag, check with Sybase Technical Support, because the trace flag may be used for purposes other than those listed here.

Table 4-1: Commonly used trace flags

Trace flag	Description	In 12.5.x?	In 15.0?	Additional remarks
291	When enabled, predicates using the form `col1 <relop> fn(col2)`—where the datatype of col2 is listed higher than that of col1—cast the expression `f(col2)` to the datatype of the lower type (in this case, col1).	Yes	No	Functionality included in Adaptive Server 15.0
333	Disables min-max optimization.	Yes	No	No longer supported
364	Uses range density instead of total density.	Yes	No	No longer supported
370	Uses min-max index as an alternative to the table scan for single-table queries. Does not perform aggregation optimization for joins.	Yes	No	No longer supported
396	Uses min-max optimization for single-table queries.	Yes	No	No longer supported
526	Prints semi-graphical execution operator tree when showplan is enabled.	No	Yes	Functionality included in Adaptive Server 15.0

If you are running with a trace flag not listed here, contact Technical Support for assistance.

Adaptive Server 15.0 discontinues the 300 series diagnostic trace flags (302, 305, 308, 310, 311, and so on), and these will be deprecated in a future release. They are replaced with showplan options, which provide better diagnostic and more readable output.

For information about tools for investigating optimizer decisions, see *Query Processing in Adaptive Server*.

Partition changes

All tables that were partitioned in earlier releases of Adaptive Server revert to a single partition during the upgrade process. Although the same commands are used to select, insert, and delete data with Adaptive Server release 15.0 (whether the table is partitioned or not), performance of your procedures may depend on the tables you are partitioning. If the tables revert to a single partition, repartition the tables.

By default, Adaptive Server uses the round-robin method to partition a table. Adaptive Server 15. 0 introduces these additional methods to partition tables:

- List
- Range
- Hash

See *What's New in Adaptive Server Enterprise* and the *Transact SQL User's Guide* for more information on partitions.

This section highlights some of the limitations of unexpected behavior when using partitions.

Partitions and primary keys and unique indexes

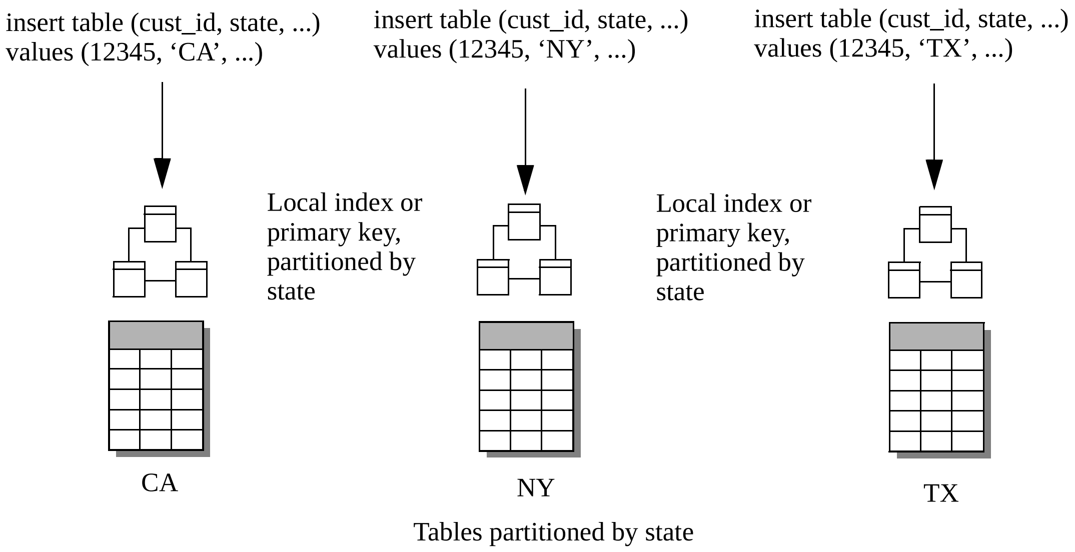
You cannot enforce unique indexes, including primary keys, if the partition keys are not the same as the local index or primary key columns (or a subset of them).

Most customers consider partitioning their tables because it allows more efficient and practical use of the parallel query feature, or because it makes database administration (DBA) tasks easier, saving time and money.

For DBA tasks, partitioning may cause unexpected behavior.. Customers typically partition a table for DBA tasks based on a date or day – for example, a modulo day number calculated as an offset, which is typically done to allow faster archiving of older data. However, sometimes this date is not in the primary key, or is only one column in the primary key. The same problems occur if the table is partitioned according to a lower cardinality division such as using a state or country and a unique key.

For example, the following table contains customers (uniquely identified by cust_id) in the United States and divides them according to sales region, and is partitioned by state.

Figure 4-1: Partitioned tables with local indexes or primary keys


insert table (cust_id, state, ...) values (12345, 'CA', ...)
insert table (cust_id, state, ...) values (12345, 'NY', ...)
insert table (cust_id, state, ...) values (12345, 'TX', ...)
Local index or primary key, partitioned by state
Local index or primary key, partitioned by state
CA
NY
TX
Tables partitioned by state

Even though you insert the same cust_id value ("12345") into the table, the insert succeeds because you insert the data into different state partitions. The index partitions act independently, and when you insert the values, you cannot know if the value already exists in another partition. For this reason, Adaptive Server displays a warning when you partition a table on a column list that does not include the primary keys, or when you attempt to create a unique local index on columns that are not used for partition keys.

For performance reasons, Adaptive Server does not enforce uniqueness. For example, in a 50-million row table, the primary key and any nonclustered index need about seven levels of indexing to find a data leaf node from the root node of the index. If you partition this table (assuming an even distribution of values), each partition has one million rows, which requires five levels of indexing. In an unpartitioned table, the unique value check takes only seven I?Os to read to the point of insertion to determine if a row with that value already exists. However, for a partitioned index, the value chech must traverse all five levels for all 50 partitions—250 I/Os total.

As a workaround, create a global unique index to enforce uniqueness instead of a local index or primary key constraint (all primary key constraints are partitioned according to the table schema). Since a global index is unpartitioned, uniqueness can still be enforced.

Multiple and composite partition keys and range partitioning

Both range and hash partitioning allow users to specify as many as 31 columns as partition keys, creating a composite partition key. For hash partitions, the partition key behaves as expected, determining how individual data rows are distributed to different partitions. However, the partition keys for range partitions can behave unexpectedly, especially when the partition keys are numeric. The unexpected behavior occurs because Adaptive Server uses the fewest partition keys possible in the sequence until it determines the appropriate partition, instead of using all the keys each time to determine where the row should be stored. The following SQL text describes the rule for determining partition keys:

```
if key1 < a, then the row is assigned to p1
if key1 = a, then
       if key2 < b or key2 = b, then the row is assigned to p1
if key1 > a or (key1 = a and key2 > b), then
       if key1 < c, then the row is assigned to p2
       if key1 = c, then
          if key2 < d or key2 = d, then the row is assigned to p2
       if key1 > c or (key1 = c and key2 > d), then
          if key1 < e, then the row is assigned to p3
          if key1 = e, then
             if key2 < f or key2 = f, then the row is assigned to p3
             if key2 > f, then the row is not assigned
```

Which is summarized as:

- If *value* is less than `key1`, use the current partition.
- If *value* equals `key1`, compare *value* to `key2`.
- If *value* is greater than `key1`, check the next partition range.

For example, if you have a table of 1.2 million customers and want to partition it to improve parallel query performance and simplify maintenance, you can partition it by fiscal quarter and customer ID, and you can archive the data every quarter, as necessary. However, if the table also includes a month and fiscal quarter columns, and since quarters occur every third month, you can use this partitioning scheme:

```
alter table telco_facts_ptn
   partition by range (month_key, customer_key)
      (p1 values <= (3, 1055000) on part_01,
      p2 values <= (3, 1100000) on part_02,
      p3 values <= (6, 1055000) on part_03,
      p4 values <= (6, 1100000) on part_04,
      p5 values <= (9, 1055000) on part_05,
p6 values <= (9, 1100000) on part_06,
p7 values <= (12, 1055000) on part_07,
p8 values <= (12, 1100000) on part_08)
```

However, instead of evenly distributing the 1.2 million rows (150,000 rows to each partition), the odd partitions contain 250,000 rows while the even partitions contain only 50,000 rows. For months one and two (January and February), when Adaptive Server compared the data values to the first key in the first partition, it was less than the key ($1 < 3$ and $2 < 3$), so it put all the January and February data into the first partition, regardless of the customer_key value. Only when Adaptive Server entered the March data did it find that the values were equal to key1 (3=3), so it needed to compare customer_key value with key2. As a result, the even partitions only have data in which the month was equal to the partition key and the customer_key value was greater than the customer_key value for the partition key before it.

Semantic partitions and data skew

In versions Adaptive Server earlier than 15.0, the segment-based, round-robin partitioning scheme supported parallel queries. However, if the partition skew exceeded a specific ratio, the optimizer considered the partitioning too unbalanced to provide effective parallel query support and process the query in serial fashion. One way to prevent this when using parallel queries is to monitor partition skews and rebalance them by dropping and re-creating the clustered index.

With semantic partitioning in Adaptive Server 15.0, data skew is no longer a consideration. Rather than evaluating the depth of the partition for parallel query optimization, the Adaptive Server optimizer considers the type of partition, the query search arguments, and so on. Because of the nature of unknown data distribution for semantic partitions, you may have data skew, but because a hash, list, or range partition signals where the data resides, the skew is unimportant.

Dropping partitions

You cannot drop a partition in Adaptive Server 15.0. To remove a partition, unpartition the table, then repartition it using the same partitioning scheme as before, but with the undesired partition left out of the DDL statement. You can also truncate all of the data in a partition.

Removing a partition can be complicated. Often, the main reason to drop a partition is because you have archived the partitioned data. You can add or drop a partition to fix minor partitioning scheme problems without repartitioning the entire table. However, doing so can lead to a problem similar to the following.

Assume you created a range-partitioned table on the partition keys of 10,20,30,40,50, and so on, archived the data in the first partition (values less than or equal to 10), and then dropped the partition, leaving partition ranges of 20,30,40,50, and so on. Later, a user inputs valid data with a partition-key value of 5. Because of the mechanics of range partitioning, the newly inserted data is successfully added to the first partition (it is less than or equal to 20). However, problems occur if you must archive some of the data (or rebalance the partitioning as more data is added), and you add a new partition with the same partition key as the original (values less than or equal to 10). The previously inserted data (the value 5) may be left stranded as local indexes all point to the "new" first partition. The best solution to this problem is to relocate the 5 rows when you add the new partition, which means you are repartitioning the table instead of adding a partition as data is relocated.

When you drop a partition, you remove all data with it, which does not cause problems. However, problems may occur when you insert new data, as described in the following example, which results in more questions about data distribution than answers.

Assume you have a hash partition on an integer column using 10 partitions. Dropping one of the partitions removes the hash bucket for 1/10th of the possible data values. Also, assume that the particular hash bucket you removed held the hash keys for integers (5,32,41,and so on). If a user inserts a value of 32, consider whether the hashing algorithm changes to reflect the full domain across the remaining 9 partitions. If so, also consider whether the purpose of dropping the partition is to redistribute the data or remove it, which means that this is actually a repartition. Perhaps, instead, the value should be rejected as the hash bucket no longer exists, which is similar to inserting an unlisted value in a list partition.

Dropping a partition can be more complex than removing a partition and all its data. Sybase opted not to include this functionality in the initial release of Adaptive Server 15.0, but may introduce this capability in a later release. However, you can repartition a table and truncate a partition, which effectively covers nonrepartitioning considerations.

Moving data in and out of partitions

You can use bcp to extract data from a partition, and you can select data out of a partition if you specify a where clause that includes only that partition's key values. You can load data directly into a partition with bcp.

You can not use select to retrieve data from a single partition by specifying the partition name.

You can move data between partitions by changing the partition-key values. If you update the column used in a partition key to a value that is in a different partition, the row is physically relocated to the other partition. Adaptive Server does this using deferred update, in which the existing row is “deleted” from the current partition, and “inserted” into the new partition. For example, if you partition a table with a state column, updating the column from NY to CA changes the partition. A deferred update consisting of the removal of the NY row and the insertion of the deleted row is recorded in the transaction log –the same as any other deferred update operation.

Changes for computed columns and function-based indexes

This section describes migration issues associated with computed columns and function-based indexes.

Computed column evaluation

If you are migrating to Adaptive Server 15.0 and considering computed columns, you must understand when computed columns are evaluated, particularly if you are trying to predetermine the likely output of non-deterministic columns. These are the evaluation rules:

- The expressions for nonmaterialized (virtual) computed columns are evaluated during query processing, so it reflects the state of the current user’s session.
- The expressions for materialized (physical) computed columns are evaluated only when a referenced column is modified.

  For example, this table has three pairs of computed columns, and each are evaluated differently:

```
create table test_table (
  rownum     int      not null,
  status     char(1)  not null,
  -- virtual columns
  sel_user   as suser_name(),
  sel_date   as getdate(),
  -- materialized columns
  cr_user    as suser_name() materialized,
  cr_date    as getdate() materialized,
  upd_user   as (case when status is not null
                then suser_name() else 'dbo' end)
                       materialized,
  upd_date   as (case when status is not null
                then getdate() else 'jan 1 1970'
end)
                       materialized
)
```

- sel_user and sel_date – are virtual columns evaluated when a user queries the table.
- cr_user and cr_date – physical and materialized columns that do not reference any other column. Their expression is evaluated only when rows are inserted. They are not affected by updates.
- upd_user and upd_date – these columns reference the status column, although the status column does not determine the value. These columns are changed only if the status column is modified by inserts and updates that set the status column to any value.

  As a result of the evaluation, the last two computed column pairs, cr_user and cr_date, and upd_user and upd_date are unaffected by queries. Although they are based on nondeterministic functions, the values are consistent for all queries.

Nonmaterialized computed columns and invalid values

The expressions for nonmaterialized computed columns are evaluated only at query time, and not during DML operations. This may lead to query problems if the formula used to create the expression is not validated before the computed column is created.

In this example, the computed column b is not evaluated until it is queried, so the domain error does not occur until the select statement runs (the error is particularly unpredictable if the select statement is embedded in a trigger):

```
create table t (a int, b compute sqrt(a))
go
insert t values (2)
insert t values (-1)
insert t values (3)
```

```
go
select * from t
go
1> select * from t
2> go
a b
----------- --------------------
2   1.414214
Domain error occurred.
```

Long identifier changes

Adaptive Server 15.0 supports longer identifiers on table names, column names, and index names: 255 bytes for regular identifiers, and 253 bytes for delimited identifiers. Because of the expanded limits, some system tables and functions now allow for longer identifiers.

Change identifier names with corresponding application changes for binding values. Make sure your existing applications are not binding names of identifiers with only 30 bytes (the previous limit), which may cause unexpected behavior, access violations, bus errors, or general protection fauls.

Error message changes

Your existing applications may expect specific error numbers or text. In Adaptive Server version 15.0, these are the error message changes that may affect your applications:

- Many messages now specify “ASE” in the error message.
- Adaptive Server raises message 12822, instead of 2714 when creating a temporary table that already exists.
- Error message 587 specifies an identity column overflow instead of error message 4916.
- When creating a Java function that does not exist in the catalogs, error message 14216 is raised instead of syntax error message 195.
- Error message 10354 is raised when a non-owner executes sp_procxmode to change the transaction mode associated with a stored procedure.
- Arithmetic overflow errors are now raised with error message 3606 with severity 16.

- Message 2579 has been replaced with message 12907 in dbcc checktable output

Open Client/SDK compatibility with Adaptive Server

This section discusses issues for using various releases of Open Client with Adaptive Server release 15.0.

New features in Open Client

How Adaptive Server and Open Client interact depends on how you determine Open Client capabilities. Specifically, “capabilities” refer to the types of requests an application sends on a specific connection and the types of server responses that a server returns for a specific connection. Before you use Open Client with the wider limits allowed by Adaptive Server 15.0, enable Open Client capabilities for these limits and modifying your code. See the *Open Client Client-Library/C Reference Manual* for information about enabling the wide-table capabilities.

Note DB-Library™ cannot use the wide limits for Adaptive Server 12.5 and later. Wider columns were introduced with Adaptive Server and Open Client 12.5. If you wrote applications with Open Client 12.5.1, they can connect to Adaptive Server 15.0. However, to take advantage of additional features in Adaptive Server 15.0, you may need to recompile with Open Client 15.0.

New features for Open Client 15.0 include:

- Wider identifiers – objects such as table names, column names, and so on now support up to 255 bytes.
- Scrollable cursors.
- New datatypes – 8-byte integer, unicode, and text.

Adaptive Server and Open Client compatibilities

The Open Client features that are available depend on which versions of the software you are running. These combinations are described below.

- Adaptive Server 15.0 and Open Client 15.0 are completely compatible. However, before you use Open Client 15.0 with Adaptive Server 15.0:

  a Relink your Open Client application with the 15.0 libraries.

  b Use CS_VERSION to establish the new version number.

  c If you are using jConnect, reconnect the drivers.

- If you are running an Open Client application earlier than 15.0 with Adaptive Server and have not relinked your application with 15.0 Open Client, the Open Client pre-release 15.0 functionality is fine, but the release 15.0 limits are not enabled. Adaptive Server sends the data according to the pre-release 15.0 limits, and truncates any data beyond these limits.

- If an Open Client 15.0 connects to an Adaptive Server, which then queries a remote Adaptive Server that is earlier than release 12.5, the remote server truncates any wide data before returning any results.

  All servers and clients in a transaction must at a minimum be at release 12.5 to use the new wide limits.

- Starting with Adaptive Server release 15.0, you must use the Sybase ODBC and OLEDB release 15.0 drivers that come with the Adaptive Server 15 SDK (instead of DataDirect).

- If your application depends on jConnect 5.5, Sybase recommends you migrate the application to jConnect 6.05 or use an existing jConnect 5.5 release area.

  - The library-naming convention has changed in Open Client 15.0. The letters “syb” are now embedded in Open Client libraries, for example, *libsybct.a*, replaces *libct.a*.

  - Open Client connections in versions earlier than 15.0, were established when a client requested a packet size (for example, using isql -A or CS_PACKETSIZE).

    With Adaptive Server 15.0, client connections can still use this method of connection, but the packet size is considered a suggestion to Adaptive Server, and Adaptive Server negotiates a packet size based on the server-configured packet size.

    No code changes or recompilation is needed, however if you need your client application to use a particular packet size, you can restrict this feature using these methods:

    - Open client – use ct_capability(CS_N)_SRVPKTSIZE).

- jConnect – specify Packetsize | Restricted maximum.
- ADO.NET/ODBC/OLEDB – specify the Connection normalized computed columns property RestrictedMaximumPacketSize.

CHAPTER 5 Making Database Administration Changes

This chapter discusses changes to Adaptive Server system administration that may cause problems if you do not prepare for them. For a comprehensive listing of changes and new features, see *What's New in Adaptive Server Enterprise*.

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. For instance, you may want to develop your own plans and scripts, or you may prefer to contact Sybase Consulting for help. Sybase Technical Support offers a Web site, the ASE Migration Resources Web page at http://sybase.com/support/techdocs/migration, for migration information and can help with technical problems such as bugs and error conditions.

Note If you perform an upgrade instead of a migration (that is, you upgrade without rebuilding your system), you will not see any change in dbid between versions of Adaptive Server. This becomes a problem only if your maintenance scripts use these IDs and you build a new system, which then uses the new ID conventions.

Upgrading from version 11.5 or earlier

You can upgrade to Adaptive Server version 15.0 only from Adaptive Server versions 11.9.x, 12.0.x, or 12.5.x.

If you are currently running Adaptive Server version 11.5, Sybase recommends that you upgrade first to version 12.0, then upgrade to 15.0.

Upgrading from version 11.9.2 or later

This section describes features included in Adaptive Server releases since 11.9.2 that may affect your upgrade to Adaptive Server version 15.0.

Optimizer changes

The query processor in Adaptive Server 15.0 is self-tuning, requiring fewer interventions than earlier versions. It also relies less on worktables for materialization since, between steps, the engine supports data flow. However, you can use more worktables when Adaptive Server determines that hash and merge operations are effective.

These are configuration parameters included with Adaptive Server 15.0 that affect optimization:

- max repartition degree
- max resource granularity
- optimization timeout limit
- optimization goal
- prod-consumer overlap factor (Adaptive Server 15.0.1 and later)
- min pages for parallel scan (Adaptive Server 15.0.1 and later)

Adaptive Server 15.0 changes the group by algorithm, and you cannot use set statistics io on with parallel plans.

Abstract plan enhancements

The abstract plan syntax for Adaptive Server 15.0 has been enhanced to support the new algorithms, which have been extended to cover optimization goal query-level setting, timeout and all set <QP algorithm> on/off/default actions (for version 15.0.1 and later).

See “Fixing queries using abstract query plans” on page 125 and the abstract plans chapters of *Performance and Tuning Guide: Optimizer and Abstract Plan* for more information.

Capturing query metrics

Adaptive Server 15.0 allows you to capture query processing metrics, aggregating, per query, the data displayed under set statistics time/io in sysqueryplans. You can access the statistical data from sysquerymetrics, including:

- The number of times metrics were aggregated for a given query
- Minimum, maximum, and average values for each elapsed time, CPU time, logical I/O cost (LIO), and physical I/O cost (PIO)

For more information, see “Using sysquerymetrics and sp_metrics” on page 111 and Chapter 5, “Query Processing Metrics” in the *Query Processor* guide.

Updating statistics after upgrading

The statistical data located in sysstatistics is upgraded during the data copy section of upgrade, which can cause rows to expand beyond the server’s page size. However, before these rows are inserted during the data copy, large statistical rows are split in two. The statistics for previously unpartitioned tables are used as statistics for the resulting partitioned tables after the upgrade, and an object’s ID is used as its partition ID in the sysstatistics..partitionid column.

Sybase strongly recommends that you run update statistics after you upgrade, particularly if you have not recently run update statistics on some tables. Because Adaptive Server 15.0 has several algorithms for sorting, grouping, unions, joining, and other operations, it is must have current statistics. Earlier versions of Adaptive Server had a single algorithm, and it was not necessary to run update statistics on fairly static data, such as data in reporting systems. However, Adaptive Server 15.0 may select a slower algorithm if it does not have current statistics, as the actual data volume far exceeds the projected volume based on the stale statistics.

Warning! Making unneeded changes to statistics may adversely affect performance. When you update statistics, consider using a larger step count than the default, particularly for large tables and when you change the histogram tuning factor. Doing this ensures that statistics on indexes on columns that contain a large number of duplicate values contain a more accurate representation of the data skew.

Running update statistics with incorrect parameters may change the number of steps in a histogram and impact performance. Apply changes first in a test environment.

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

Automatic update statistics

In Adaptive Server 12.5.1, Sybase provided the capability to automate running update statistics using the Job Scheduler utility program. Adaptive Server 15.0 improves this feature by allowing database administrators to set thresholds before the Job Scheduler runs update statistics on a specific table or partition. This threshold control ensures that update statistics is run only when needed, reducing the time needed for maintenance operations.

See the *Job Scheduler's User's Guide* for more information on setting up the Job Scheduler engine and the Sybase Central interface.

"Automatically updating statistics" does not mean that index statistics are automatically maintained by Adpative Server and that update statistics is no longer necessary. Users frequently request that the server track the statistics modifications for each DML statement, eliminating the need to run update statistics entirely. However, doing so may slow down OLTP operations, even if it is done outside the scope of the transaction. It is likely that multiple users would attempt to modify statistics for the same rows in systabstats, and the contention caused by these attempts would result in effectively single-threading the system.

In addition, incrementally adding rows to a large table might not accurately update the statistics because the newly computed density masy have the same value due to precision loss. For example, if you add 100,000 rows to an existing 1,000,000-row table containing a date column based on the date the row was added to the table (that is, it defaults to getdate()), and you add the rows one at a time, the range cell density would not change because each new row as it is added only amounts to 1/1,000,000th of the table – or .000001. However, 100,000 rows adds 10% to the overall table size.

datachange function

Automatic update statistics is based on the datachange function, which returns the percentage of data modified within a table. You can include this function in existing update statistics scripts to take advantage of its tracking abilities. For example:

```
select @datachange = datachange("authors", null, null)
if @datachange > 50
begin
    update statistics authors
end
```

Consider the following when you use the datachange function:

- The percentage datachange returns is based on the number of DML operations and the table size. However, each deferred operation counts as two separate operations, one delete and one re-insert. Consequently, when updating multiple records in the same statement, datachange may report a percentage up to twice as high as the actual number of rows modified.
- The datachange parameters are *table_name*, *partition_name*, and *column_name*, in that order. This allows you to detect the change in particularly volatile fields or a specific partition, and update the index statistics for specific indices as opposed to detecting the change in all indices.

- datachange reports a percentage changed instead of the number of rows changed because reporting the number of rows does not really provide useful information by itself and is useful only when compared in the context of the size of the table. For example, a value of `datachange=5,000` could be significant if the table contains 5,100 rows, but insignificant if it contains 500 million. By using a percentage, it makes it easier to establish relative thresholds in maintenance scripts.

update statistics on partitions

After you migrate to Adaptive Server 15.0, running update statistics on a specific partition should reduce the time you need to run update statistics in general. Most large tables that take a long time to run update statistics contain a lot of historical data, but few, if any, of these historical rows change, yet update statistics must scan them all. Additionally, when viewed from the perspective of the entire table, even one million rows added to a 500 million row table is only a 0.2 percent change, which suggests that statistics do not need to be updated. However, because these are likely the rows most often used and their distribution heuristics are not included in the range cell densities, it is likely that query optimization suffers.

If you partition the data, you can skip older, static data when you run update statistics after the first time, especially if you partition on a date range. After this first run, use datachange to check the amount of change within the current partition and run update statistics only as necessary. All partitions are named, and if you did not supply one (that is, you used the hash partition syntax), Adaptive Server provides a default name for you, similar to tempdb tables. Use sp_help to identify the system supplied names.

This example is based on a system-supplied name and focuses on the p_partkey column:

```
select datachange("mytable","part_1360004845", "p_partkey")
go
---------------------------
100.000000
```

This is a good candidate for running update statistics on the part_1360004845 partition.

This example allows update statistics to focus on a specific partition (and, in this case, column):

```
update statistics mytable partition part_1360004845 (p_partkey)
go
```

The reduction in time for update statistics may allow you to create statistics or heuristics on columns that you previously avoided due to time constraints. For example, this example creates a histogram on col1 and densities on the combined col1,col2 pair. :

```
update statistics mytable (col1, col2)
```

To update statistic by partition and create a histogram on col2:

```
update statistics mytable partition part_1360004845 (col1, col2)
go
update statistics mytable partition part_1360004845 (col2)
go
```

Changes to functions

Adaptive Server 15.0 deprecates the system functions that it used to report space usage in previous versions and has replaced them with partition-aware versions. Table 5-1 lists these functions, along with the syntax changes.

Table 5-1: List of function changes for Adaptive Server 15.0

Function and syntax in 12.5	Function in 15.0
data_pgs(object_id, {doampg | ioampg})	data_pages(dbid, object_id [, indid [, ptn_id]])
used_pgs(object_id, doampg, ioampg)	used_pages(dbid, object_id [, indid [, ptn_id]])
reserved_pgs(object_id,{doampg | ioampg})	reserved_pages(dbid, object_id [, indid [, ptn_id]])
rowcnt(sysindexes.doampg)	row_count(dbid, object_id [, ptn_id])
ptn_data_pgs(object_id, partition_id)	(data_pages())

Adaptive Server 15.0 replaces the OAM page parameters doampg and ioampg with the more user-friendly indid (index ID) and ptn_id (partition ID) parameters. Adaptive Server also changes how these functions are used.

In Adaptive Server 12.5, these functions were used in queries involving the sysindexes table because space allocations were tracked with sysindexes and consequently, the sysindexes doampg and ioampg columns provided the OAM page parameters. For example, a common query used in Adaptive Server 12.5 to calculate total space used for each of the nonclustered indexes on a particular table was similar to:

```
-- ASE 12.5 logic to report the spaced used by nonclustered indices
select name, indid, used_pgs(id, doampg, ioampg)
   from sysindexes
   where id=object_id('salesdetail')
```

```
     and indid > 1
```

In Adaptive Server 15.0, space is linked to syspartitions instead of sysindexes, so the query above is rewritten for Adaptive Server 15.0 as:

```
-- ASE 15.0 logic to report the spaced used by nonclustered indices
select i.name, p.indid, sum(used_pages(db_id(), p.id ,p.indid))
   from sysindexes i, syspartitions p
   where i.id=object_id('salesdetail')
     and p.id=object_id('salesdetail')
     and i.indid > 1
     and p.indid > 1
     and p.id=i.id
     and p.indid=i.indid
   group by i.name, p.indid
   order by p.indid
```

The deprecated Adaptive Server 12.x functions still execute, but return a value of 0; they rely on sysindexes.doampg and sysindexes.ioampg, which Adaptive Server no longer maintains. syspartitions has similar structures in the columns datoampage and indoampage, but the values for these columns are on a partition basis, so you must aggregate index space usage for partitioned tables.

Space reporting system function

Table 5-1 lists the changed functions, along with the old and new syntax. These changes do not affect end-user application, but may impact some or your scripts and utilities.

The change that may have the biggest effect on your scripts is the replacement of sysindexes.doampg or ioampg with sysindexes.indid and partition_id. In earlier versions, the functions used scans of sysindexes, but in Adaptive Server 15.0, they use scans of syspartitions (often joined with sysindexes). For example, in Adaptive Server 12.5, this is used to report the space used by nonclustered indices:

```
select name, indid, used_pgs(id, doampg, ioampg)
from sysindexes
where id=object_id('authors')
and indid > 1
```

In Adaptive Server 15.0, this is the syntax to report the space used by nonclustered indices:

```
select i.name, p.indid, used_pages(dbid(), p.id ,p.indid)
from sysindexes I, syspartitions p
where i.id=object_id('authors')
and i.indid > 1
and p.indid > 1
and p.id=i.id
and p.id=object_id('authors')
and p.indid=i.indid
order by indid
```

The script is different because storage in Adaptive Server 15.0 is linked to syspartitions instead of sysindexes in earlier versions.

This example reports the spaced used by nonclustered indexes by partition in Adaptive Server 15.0:

```
select p.name, i.name, p.indid, used_pages(dbid(),
p.id, p.indid, p.partitionid)
from sysindexes I, syspartitions p
where i.id=object_id('authors')
and i.indid > 1
and p.indid > 1
and p.id=i.id
and p.id=object_id('authors')
and p.indid=i.indid
order by p.partitionid, p.indid
```

Changes to system tables

Sybase has updated the system tables for Adaptive Server 15.0 to reflect the changes for semantic partitions and encrypted columns, adding new system tables and extending others to include partition information, and adding new objects to the object classes. These changes may affect third-party tools and custom scripts.

See *What's New in Adaptive Server Version 15.0?* for more information.

Changes for disk space allocations

Although earlier versions of Adaptive Server reported disk space allocations in sysindexes, Adaptive Server 15.0 reports disk space allocations in syspartitions. Table 5-2 describes the space pointers for earlier releases of Adaptive Server and their equivalents in syspartitions in Adaptive Server 15.0.

Table 5-2: Space pointers in different versions of Adaptive Server

Type of space association	Column names in sysindexes in earlier versions	15.0 syspartitions equivalent
Unique row	id + indid	id+indid+partionid
First page	first	firstpage
Root page	root	rootpage
Data OAM page	doampg	datoampage
Index OAM page	ioampg	indoampage

You must change custom DBA scripts or previous dbcc commands that used these locations to reflect the current implementation. Sybase has already modified the published dbcc commands. If you used previously undocumented dbcc commands (such as dbcc pglinkage()), these may fail because they have been deprecated or simply not maintained. If a data corruption problem occurs, contact Sybase Technical Support for the correct procedures for Adaptive Server 15.0. Continuing to use undocumented dbcc commands from earlier versions may cause corruptions.

Changes to third-party tools

Upgrading to Adaptive Server 15.0 should not affect third-party applications or in-house applications that access the database for normal DML and query operations. However, system table and function changes may cause problems for third-party tools or custom applications that perform DDL or access the system catalogs.

For example, previous releases of Embarcadero's DBArtisan that were compatible with Adaptive Server 12.5 included a widely-used feature in which the object browser displayed the table names along with the number of rows in the table. This row count was derived by using the rowcnt function, which has now been deprecated. As a result, if you use these versions of DBArtisan, you may be surprised to see that your 15.0 server is "missing all the data," as DBArtisan reports 0 rows for every table, in addition to a large number of warnings. Although Sybase allows the deprecated function for compatibility reasons, a warning message is returned.

Contact your thirdparty utility vendors to see when they are releasing a version compatible with Adaptive Server 15.0.

Changes to database IDs

In Adaptive Server version 12.0 and later, the dbids of the following databases start at 31513:

- dbccalt
- dbccdb
- sybsecurity
- sybsystemdb
- sybsystemprocs

If you drop and re-create these databases after an upgrade, Adaptive Server applies these new dbids. The dbids for all other databases are determined in the same way as in earlier versions.

ASE plug-in for Sybase Central

This section describes issues associated with Sybase Central version 4.3.

Note You must have Java installed to run Sybase Central.

Adaptive Server 15.0 includes Sybase Central release 4.3. However, this version of Sybase Central may not be the same build as Sybase Central 4.3 installed from other Adaptive Server installations. The Sybase Central build shipped with Adaptive Server 15.0 supports plug-ins for SySAM, Unified Agent Framework (UAF), and includes new functionality not available in previous releases (starting, stopping, pinging remote servers, remote error log viewing, automatic server detection, server groups, and so on.

The previous release of Sybase Central was located in *$SYBASE*. The version shipped with Adaptive Serer 15.0 is located in *$SYBASE/shared/sybcentral43*. (*%SBASE%\Shared\Sybase Central43* on Windows). This change of location can lead to a number of problems if you leave the previous version intact as program launch icons, or if your *path* settings or CLASSPATH points to the previous location.

Rename your old Sybase Central directory to disassociate it from the new version. After you have used the new version for a period of time and you are sure all the product plug-ins are compatible with the new release, you can delete the old version. Product plug-ins are individually available from www.sybase.com.

Displaying servers listed in sql.ini

The Adaptive Server plug-in to Sybase Central no longer displays all servers listed in the *sql.ini* file. Instead, Sybase Central lists only those servers that you connected to earlier, or those servers that are started as Windows NT services. To access a new server for the first time, use the Connect menu option in the Adaptive Server plug-in to select a server listed in the *sql.ini* file.

Troubleshooting

When using Sybase Central:

- Make sure you are using the most current build (4.3.0.2419 as of ASE 15.0 ESD1). You may be launching an old version of Sybase Central (for example, version 4.1) or an older build of version 4.3 that is not compatible with the Adaptive Server 15.0 plug-in. Make sure that you are using the current build of Sybase Central, and that you are launching the version from *%SYBASE%\Shared\Sybase Central 4.3*. You can verify you are launching this version by opening a DOS window, navigating to this directory, and executing *scjview.bat*.
- Sybase Central with the UAF plug-in uses Java security policies. However, VPN software uses TCP redirection, such as VSClient from InfoExpress. If you find these to be incompatible, exit the VPN client application and try running Sybase Central again.

When it crashes, Sybase Central creates a stack trace in a file named *scj-errors.txt* (if there are multiple crashes, the files are numbered sequentially, so *scj-errors-2.txt* is the next file, and so on) located in the Sybase Central home directory. If you are reporting problems to Sybase Technical Support, include this file because it identifies the plug-in and Jar file versions used by Sybase Central and the plug-ins.

Interactive SQL

The Adaptive Server plug-in includes Interactive SQL. Interactive SQL which is a common client utility for all the Sybase servers and can connect to Open Server-based applications such as Replication Server. As a Java-based client, Adaptive Server plug-in can run on Linux and other UNIX platforms and is a more advanced product than the Jisql utility included with earlier releases of Adaptive Server.

Interactive SQL allows you to execute SQL statements, build scripts, and display data to the server. You can use it to:

- Browse the information in a database
- View a graphical representation of your query

- Edit your result sets, and have the changes update the database
- Export to MicroSoft Excel and other formats
- Test SQL statements that you plan to include in an application
- Load data into a database and carry out administrative tasks

Interactive SQL can run command files or script files. For example, you can create repeatable scripts to run against a database and then use Interactive SQL to execute these scripts as batches.

Correcting errors in batch scripts

The batch scripts for Adaptive Server 15.0 and ESD #1 contain an error in their call to the Interactive SQL class file. In some scripts, the -Dpath option is incorrect, although the *%path%* value for the option is present.

To correct this issue, move to *%SYBASE%\dbisql\bin* and edit the *dbisql.bat* file and make sure the execution line reads shown below:

Note The line breaks are for document formatting; this command is actually one continuous line. Pay particular attention to the dashes (“-”) —which are switch characters—when copying and pasting this line into your script.

```
"%JRE_DIR%\bin\java" -Disql.helpFolder="%DBISQL_DIR%\help" -
Dsybase.jsyblib.dll.location="%SYBROOT%\Shared\win32\\" -
Djava.security.policy="%DBISQL_DIR%\lib\java.policy" -Dpath="%path%" -
classpath
"%DBISQL_DIR%\lib;%isql_jar%;%jlogon_jar%;%jodbc_jar%;%xml4j_jar%;%jconn_j
ar%;%dsparser_jar%;%helpmanager_jar%;%jcomponents_jar%;%jh_jar%;%jsyblib_j
ar%;%planviewer_jar%;%sceditor_jar%;%uafclient_jar%;%jinicore_jar%;%jiniex
t_jar%;%jmxremote_jar%;%jmxri_jar%;%commonslogging_jar%;%log4j_jar%"
sybase.isql.ISQLLoader -ase %*
```

You can also trim the *%path%* environment variable to only what is necessary. For example:

```
set PATH="c:\sybase\ASE-15_0\bin;c:\sybase\OCS-15_0\bin;.;"
```

SQL Advantage no longer supported

Sybase no longer maintains SQL Advantage™. It connects to Adaptive Server 15.0, but because it does not include some newer API features in Open Client 15.0, it has limited functionality.

If you have problems running SQL Advantage on a machine that includes Adaptive Server 15.0 PC-Client, SQLAdvantage may be finding Open Client 15.0 in the path ahead of Open Client 12.5. Because Open Client 15.0 renamed some of the *dll* libraries, SQLAdvantage fails when it uses these libraries. As a workaround, create a shell script that includes only the Open Client 12.5 library directories (and none of the Open Client 15.0 library directories) in the path environment variable. Launch SQLAdvantage from this shell script.

sybsyntax

The sybsyntax utility, which provides online help and Transact-SQL syntax at the isql command line, was removed from Adaptive Server version 12.0, but has been added as part of Adaptive Server version 12.5. Install the latest version of sybsyntax to view the most recent syntax.

Changes to documentation

The following changes have been made to the Adaptive Server documentation set:

- *Managing and Monitoring Adaptive Server Enterprise* has been discontinued.
- The *Utility Guide* is now a generic book that includes all utilities, regardless of platform.

sybsystemdb

The master database is used for the spt_values table and two-phase commits. The database is automatically created on the master device by the configuration utility.

Because this database may have a lot of activity and use a lot of log space, Sybase strongly recommends that you either create sybsystemdb on another device before you upgrade or that you move it to another device after you upgrade. This protects the master device in case the sybsystemdb log fills up.

If you do not plan on using the features that require sybsystemdb, you can keep it at the minimum size.

bcp* with *syslogins

If you try to use bcp to copy data into syslogins from an earlier version of Adaptive Server, it may fail due to additional columns that were added in Adaptive Server 12.5.

To use bcp to copy data into a 12.5. version of syslogins, you can create a dummy table to hold the data, add two columns to this dummy table, and then copy the data into the 12.5 syslogins.

Note This issue may also occur with other system tables that have added columns. You might encounter the problem first with syslogins if you need to copy this data as part of the upgrade.

See *What's New in Adaptive Server Enterprise?* for changes to system tables in Adaptive Server releases.

Maximum number of users and logins

The maximum number of logins to the server and users to the database was increased in Adaptive Server version 12.5:

- Number of logins – 2 billion plus 32K
- Number of database users – 2 billion less 1032193
- Number of groups in a database – 1032193

Because of these changes:

- You can use negative values for user IDs (uid)
- The server user ID (suid) associated with a group or a role in sysusers is no longer equal to the negation of their uid. In Adaptive Server version 12.0, every suid associated with a group or a role in sysusers is set to -2(INVALID_SUID).

See *What's New in Adaptive Server Enterprise?* for ID ranges.

New reserved words

See "Reserved words" on page 38 for information about reserved words.

Configuration parameters

Any configuration parameter values that you have set are preserved by the upgrade process. Only those parameters that are set at "DEFAULT" in the server configuration file are changed to the new version's default value. Check that your parameter settings are not lower than the new default values in Adaptive Server 15.0. The following parameters may cause problems during or after the upgrade if they are set too low:

- stack size
- cpu grace time
- enable housekeeper GC

stack size

The default value of the stack size parameter has increased over the last several releases. While stack size varies from platform to platform, the changes on Solaris (shown in the table below) illustrate the increase in stack size requirements with successive versions:

Server version	Minimum stack size
11.0.x	24576 (24K)
11.5	34816 (32k)
11.9.2	34816 (32k)
12.0	46090 (45K) – 32-bit 86016 (84K) – 64-bit
12.5	46090 (45K) – 32-bit 86016 (84K) – 64-bit
15.0	46090 (45K) – 32-bit 86016 (84K) – 64-bit

Using a stack size that is set too low can cause a stack overflow during upgrade or later, during processing.

Check your stack size parameter and reset it to default, if necessary. You can set the stack size configuration parameter by replacing the existing value in the configuration file with the word "DEFAULT".

For more information on configuration parameters, see the *System Administration Guide, Volume 1*. For a comprehensive list of new and changed configuration parameters, see *What's New in Adaptive Server Enterprise?*

Increased memory

Adaptive Server 15.0 uses more memory than previous releases, allocating more memory for the data server, procedure cache, and named caches. Many system procedures in this release now call other procedures. Query optimization in Adaptive Server 15.0 is much more complex than in earlier releases, so sites with minimally configured procedure caches are likely to run out of space.

For example, Adaptive Server 15.0 uses in-memory sorting and grouping algorithms, and requires additional procedure cache space for the auxiliary scan buffers used to track sort buffers and additional memory for sorting. Since Adaptive Server 15.0 uses in-memory sorting instead of worktables, the additional memory requirement is in whichever data cache tempdb is bound.

Although it is impossible to predict how much additional memory is required, the additional requirements are probably only a small percentage above what you currently allocate (somewhere between 5 and 25 percent). Carefully monitor memory usage after upgrading. Use sp_helpcache to check cache sizes before and after the upgrade and adjust memory, if necessary, to restore the original value of the default data cache after the upgrade is complete.

See the *System Administration Guide, Volume 2* for information on configuring memory and data caches; see the *Performance and Tuning Guide: Basics* for more information on memory issues.

Changes that affect upgrade and server functionality

The following changes in Adaptive Server version 15.0 affect upgrade and server functionality:

- System catalogs use row "anchors" that are stored in a per-database pseudo-catalog named sysanchors. Like sysgams, sysanchors is not a normal database table and cannot be queried.
- Most system catalogs are row-locked.
- An object's partition ID is unique to that object.

The following are additional issues that you should consider when you migrate to Adaptive Server 15.0 and later:

- What Adaptive Server called “partitions” in previous releases are now called “slices,” and are obsolete. When you upgrade from version 12.5.x and earlier, Adaptive Server 15.0 and later changes all database tables, including the slice-partitioned tables supported in version 12.5.x, into unpartitioned tables. Indexes do not change; they remain global and unpartitioned. If you want partitioned tables, you must repartition them manually after the upgrade.
- During migration syspartitions is renamed sysslices. This data is used only during upgrade, after which it is removed. sysslices remains as an empty table after the upgrade process completes.
- This release uses more procedure cache than previous releases. The amount of procedure cache varies with the application, but generally, Adaptive Server 15.0 uses appoximately 10% more than previous releases.
- Adaptive Server restricts you from running load tran commands that reissue a create index that require a redo of a sort from older Adaptive Server releases. Adaptive Server recovers logs containing this operation up to the point that the create index first occurs, and invalidates subsequent load tran commands. You cannot run another load tran command until you first run a load database command.

Device size

As of version 15.0, Adaptive Server supports more than 2 billion database devices, each of which can be 2,147,483,648 2K-blocks, that is, up to 4 terabytes per device. You can configure as many of these devices as available memory supports.

bigint support

Adaptive Server 15.0 includes the bigint exact numeric datatype.

Table 5-3 shows the range of numbers allowed by the bigint datatype.

Table 5-3: Ranges for big int datatype

Datatype	Range of signed datatypes
bigint	Whole numbers between -2^{63} and $2^{63} - 1$ (from -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807, inclusive)

As part of the bigint datatype, Adaptive Server now also includes the hextobigint, biginttohex, and count_big functions. For more information, see the *Reference Manual: Blocks*.

unsigned int support

The unsigned integer datatypes (for example, unsigned int and unsigned smallint) allow you to extend the range of the positive numbers for the existing integer types without increasing the required storage size. Unsigned datatypes cannot store negative numbers.

Integer identity

Adaptive Server version 15.0 allows you to use the following datatypes as identity values:

- bigint
- int
- numeric
- smallint
- tinyint
- unsigned bigint
- unsigned int
- unsigned smallint

Separate device ID column

In earlier releases, the device number (the *vdevno*) was the high-order byte of the 32 bit page implementation. Finding the *vdevno* often meant that you performed complicated calculations with values like 2^{24} to isolate the high-order byte. The technique also included an implied limit of 255 devices. If you attempted to associate device fragments from master..sysusages with master..sysdevices, you had to join the tables using a between clause based on the high and low virtual page numbers.

In Adaptive Server 15.0, the virtual page number is now two 32-bit integers, one for the device number (*vdevno*) and one for the page ID itself. The *vdevno* is included in sysusages and sysdevices. Because of these changes, you must modify your scripts that calculate space consumption. For example, this script written for Adaptive Server 12.5:

```
select d.name, u.size
from sysuages u, sysdevices d
where u.vstart >= d.low
and u.vstart <= d.high
and u.dbid = <database id>
```

Has been rewritten for Adaptive Server 15.0 as:

```
select d.name, u.size
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and u.dbid = <database id>
```

File system or raw partition?

When deciding whether to use file systems or raw partition devices, there are a number of factors you should consider, such as operating system file system implementation, application I/O profiles, available operating system resources such as memory and CPU, operating system tuning for file system cache limits, and swapping preferences.

For earlier releases of Adaptive Server, file system devices were good for large read operations where the file system read-ahead outpaced Adaptive Server asynchronous prefetch capabilities, however, raw partitions performed better for write activity, particularly in high concurrency environments.

Adaptive Server 12.0 included the device dsync parameter, which implemented dsync I/O for file system devices, enabling updates to the device to take place directly on the storage media, or buffering them with the UNIX file system. Although it may have appeared that dsync bypassed the file system buffer to ensure recoverability, it still used the file system buffer, but forced a flush after each file system write. This double buffering in both Adaptive Server and the file system cache—plus the flush request—caused slower response times for writes to file system devices than for raw partitions.

Adaptive Server 15.0, includes the directio parameter for disk init, disk reinit, and sp_deviceattr, which implements direct I/O, improving the slower write times by allowing you to configure Adaptive Server to transfer data directly to disk, bypassing the operating system buffer cache. These platforms support the directio parameter:

- Sun Solaris
- IBM AIX
- Microsoft Windows

Considerations before using directio

Consider the following before you use directio:

- You may have to tune your operating system kernel, mount the file system with special options (for example, the forcedirectio mount option on Solaris), and make sure the operating system patch levels are sufficient for a high volume of direct I/O activity.
- The directio and dsync parameters are mutually exclusive. If you are currently using the dsync parameter, but you want to use directio, you must first disable the dsync parameter, enable directio. However, changing a device attribute requires that you restart Adaptive Server.
- Make sure that the memory requirements for file system caching and CPU requirements for changes in operating system processing for I/O requests do not affect the Adaptive Server resources.
- Test how your application scales for user loads of 25%, 50%, 75%, and 100% before you switch a device from using raw, dsync, or direct I/O. By testing for each of these scenarios, you can see if the scaling is linear or degrades as the use load gets closer to 100%. If performance flattens at a point, but you still need to increase the user load because of increased user population, you may need to add more resources to Adaptive Server.

- If you are running Adaptive Server with tempdb on a file system on SMP hardware, you may get more benefit from enabling directio than disabling dsyncio. However, test your applications at a variety of load levels with different tempdb device configurations. You may find that disabling dsyncio has a greater advantage on smaller SMP systems with lower user concurrency in tempdb or with fewer, but larger temporary tables, while enabling directio has a greater advantage on larger SMP systems or on systems with high concurrency in tempdb with smaller, but higher write activity, tempdbs. It may be beneficial to have one tempdb running with file system devices and dsyncio disabled for nightly batch reports, but to use raw partitions or file system devices with directio enabled, for the OLTP processing temporary databases during the day.

  Enabling dsyncio may dramatically degrade query performance for sorting operations (for example, adding an order by clause to your query, or if you have dsyncio enabled for tempdb).

  Generally, if you use sorting operations, disable dsyncio, or enable directio.

  This example creates tempdb on a new device with dsyncio disabled:

```
USE master
Go
DISK INIT  name = 'tempdbdev01', physname = '/tempdb_data' , size = '4G',
dsync = 'false'
Go
DISK INIT name = 'tempdblogdev01', physname = '/tempdb_log', size = '4G',
dsync = 'false'
Go
ALTER DATABASE tempdb ON tempdbdev01 = '4G'  LOG ON tempdblogdev01 = '4G'
Go
USE tempdb
Go
EXEC sp_dropsegment 'logsegment', 'tempdb', 'master'
go
EXEC sp_dropsegment 'system', 'tempdb', 'master'
go
EXEC sp_dropsegment 'default', 'tempdb', 'master'
go
```

  If you already have devices established for tempdb, you need only to disable dsyncio, but you will need to restart Adaptive Server:

```
EXEC sp_deviceattr 'tempdbdev01', 'dsync', 'false'
Go
EXEC sp_deviceattr 'tempdblogdev01', 'dsync',
'false'
```

Row-locked system catalogs

In Adaptive Server 15.0, system catalogs, except the message tables, fake tables (non-row-oriented tables), and logs – are row-locked. These tables no longer have a clustered index, but instead now have a “placement” index, with a new index ID. Pages at the data level for Adaptive Server version 15.0 are not chained together, and table starting locations are no longer set, but are now randomly generated.

Databases are larger

Adaptive Server 15.0 uses larger databases because it uses row-locked system catalogs and adds several new system catalogs.

The new minimum sizes are:

- Minimum database size – 6 allocation units (1536 pages, 3MB on a 2K page)
- Minimum master database – 26 allocation units (6656 pages, 13MB on a 2k page)
- Minimum tempdb size – the larger of 4MB or 6 AU
- Minimum sybsystemprocs size – 124MB (132MB recommended)

The default database size configuration parameter default value has changed:

- 2K pages – 3MB
- 4K – 6MB
- 8K – 12MB
- 16K – 24MB

Make sure the database you are upgrading has plenty of free space. Although the preupgrade utility calculates the space necessary for catalog changes, be generous with the amount of available space in a database when performing an upgrade. Try to have at least 25% of the space free for databases less than 10GB. Also make sure the transaction log is as clear as possible. Truncate the transaction log after performing a full database dump, because the catalog changes also require log space.

Before upgrading, make sure the master database is using only about 50% of the available disk space and that sybsystemprocs has sufficient disk space for new stored procedures. Make a dump of the master database and dump the transaction log to truncate it, which is often the cause of space consumption in the master database.

Utilities like DBExpert, use abstract query plans, which are captured in the system segment. Make sure you have enough free space in the system segment beyond the catalog expansion requirements. Adaptive Server 15.0 adds the sysquerymetrics feature, which uses system segment space. Before you upgrade, prepare for sysquerymetrics by expanding the existing system segment to other devices if it was restricted to a smaller disk or a disk with little free space.

#temp table changes

In versions earlier than 15.0, the 30-character limit for #temp table names meant that user temporary tables in tempdb were named with the hash symbol (#), 12 distinct characters, plus a 170-byte hash. Names shorter than 12 characters were padded with underscores to achieve a length of 12 characters.

This example creates a temporary table named
#temp_t1______0000021008240896:

```
select *
into #temp_t1
from mytable
where …
```

Adaptive Server 15.0 names for temporary tables:

- Are not padded with underscores
- Do not have a limitation for distinct characters
- Do not have a limit of 30 characters for object names

This change should not affect applications, except applications built in Adaptive Server 15.0 may not be backwardly compatible to Adaptive Server 12.5.

The following examples describes successful and failed scenarios for creating temporary tables in Adaptive Server versions 12.5 and 15.0.

This example fails in Adaptive Server 12.5, but succeeds in Adaptive Server 15.0 because in 12.5, the automatic padding with underscore results in tables with the same name:

```
create table #mytemp (…)
create table #mytemp___ (…)
```

This example fails in Adaptive Server 12.5 because the temporary table names are truncated to 12 characters. It succeeds in Adaptive Server 15.0 because this version does not truncate table names:

```
create table #t12345678901 (…)
create table #t1234567890123 (…)
```

This example refers to the same table in Adaptive Server 12.5, but to different tables in Adaptive Server 15.0 for the same reason (name truncation) described in the example above:

```
select * from #t12345678901
select * from #t1234567890123456
```

New limits for regular delimited identifiers

In Adaptive Server version 15.0, delimited identifiers are enclosed in double quotes (“) or brackets ([ ]), and variables can be 254 bytes, since the @ counts as 1 byte.

Extended identifiers apply to:

- Table names
- Column names
- Index names
- View names
- User-defined datatypes
- Trigger names
- Default names
- Rule names
- Constraint names
- Procedure names
- Variable names

- JAR names
- Function names
- Names for time ranges
- Name of light-weight process (LWP) or dynamic statements
- Application context names

These system tables have changed because of long identifiers:

- sysattributes
- sysaudits01-08
- syscacheconfig
- syscolumns
- sysconfigures
- sysindexes
- sysjars
- sysobjects
- sysprocesses
- systimeranges
- systypes

These are possible issues you may encounter because of longer identifier names:

- Increased stack usage
- Increased use of disk space.
- You may need to change your user-defined stored procedures or applications to accommodate the longer identifiers.

SySAM license manager

The SySAM license manager enforces Sybase licensing. It requires a valid license for the platform for both base functionality and any options you have purchased. SySAM also ensures that customers are entitled to receive EBFs. Each license is tied to a specific Adaptive Server host or license server host.

Adaptive Server 15.0 with SySAM 2.0 can co-exist with Sybase products that use older SySAM technology.

For more information about SySAM, see the *Installation Guide* and the *Configuration Guide* for your platform, or the SySAM Web site: http://www.sybase.com/sysam.

buildmaster obsolete

Adaptive Server releases later than 12.5 do not use the buildmaster utility. Instead, this functionality is included in the dataserver -b option (sqlserver.exe on Windows), which runs the build mode. The dataserver command allows you to create master devices and databases with logical pages of size 2K, 4K, 8K, or 16K.

To create a new Adaptive Server, issue dataserver using the -b and -z options. For example, to build a 100MB master device using the default logical page size (2K) and start the server, issue:

```
dataserver -d /var/sybase/masterdb.dat -b100M -sMASTER2K
```

To build a 100MB master device with a logical page size of size 4K:

```
dataserver -d /var/sybase/masterdb.dat -b100M -z4K -sMASTER4K
```

For more information about the dataserver utility, see the *Utility Guide*.

Warning! Because of changes in the master device and database, as well as the elimination of the buildmaster command, recovery is more complex if you lose your master device or database following an upgrade. Before upgrading, make both a physical dump and a bcp copy of the master database.

CHAPTER 6

Ensuring Stability and Performance

This chapter helps you evaluate testing methods and develop a testing plan.

Overview

The primary goals of testing is to ensure that, after migration:

- Application behavior is predictable.
- Application and operational service levels are preserved or exceeded.
- Test and production systems are stable and the data is safe.
- The upgrade has been successful and has not adversely impacted the production system.

Setting up the test environment

Ideally, set up a dedicated hardware configuration (including subnets) and Adaptive Server exactly like your production system. Creating an identical system lets you make valid comparisons, perform real tuning as part of the migration effort, and if you choose to do so, switch the test system to production later on. Create the test system as described in these sections:

- Make backups
- Use scripts to create the test system
- Create your databases by loading backups
- Install monitoring tables
- If your test environment is not an exact duplicate

Make backups

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

Use scripts to create the test system

Using the object creation scripts you gathered, wrote or reverse-engineered in Chapter 2, "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 you do not intend to rebuild your production environment, you may prefer to create your databases as described in "Create your databases by loading backups" on page 91.

Run dbcc commands in the test databases immediately after creating them to be sure there are no problems.

Install monitoring tables

Monitoring tables (also known as MDA tables) are available in Adaptive Server version 12.5.x and later.

After you configure Adaptive Server for monitoring tables, Adaptive Server uses the monitoring tables and sysquerymetrics (particularly after upgrade) to find affected queries as well as changes in system resource consumption.

Configuring the monitoring tables consists of creating a loopback server entry, adding mon_role privileges, and installing the monitor tables from the most current *installmontable* script. If you previously installed the monitor tables from an earlier release, then applied any EBFs, you may need to reinstall the tables.

Along with the monitoring tables, the sysquerymetrics system table and sp_metrics may be helpful for identifying under-performing queries.

For more information about the monitoring tables, see the *Performance and Tuning Guide: Monitoring*.

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.

1 Create the database using the for load option of the create database command.

2 Load the backups you made from the production database.

3 Issue the online database command which automatically upgrades the database if it is not at version 15.0.

For command syntax, see the *Reference Manual: Commands*.

If your test environment is not an exact duplicate

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

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. You can validate functions not tested at this time during user acceptance tests.

Establishing performance criteria

Depending on your migration plan, 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*:
  - Measure the overhead of the replication mechanism. For example, if the replication costs 10%, do not begin parallel operations until release 15.0 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*, aim 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 3, “Writing a Migration Plan” for information on migration methods.

Developing fallback procedures

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

During simple testing cycles, it may be faster to back out unwanted changes. However, Sybase does not recommend this for *timed runs* during benchmarking. 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 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 re-create your environment if necessary.

Summary of testing techniques

You can use a variety of testing techniques in your test plan. Table 6-1 summarizes the advantages and disadvantages of various testing techniques and tools:

Table 6-1: Testing techniques' advantages and disadvantages

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 • Back-end focus may help locate the cause of a problem	• Only tests backend server • 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
Concurrency and capacity testing	Use third-party load-testing tools	• Tests both front and back ends • Language and looping constructs make these tools	• Heavy processing requirements for test tool—may even require additional hardware, or else results may be skewed • Learning and development curve to write multiuser test simulations • Risk of bugs in test harness may skew results • Dependent on strong analysis of process/transaction profiles

Technique	Description	Advantages	Disadvantages
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	Use 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 must write special benchmark scripts, rather than rewriting applications as benchmarks.

To write the benchmark script:

- Add a function to funcs.c for each transaction.
- Generate any runtime data required (such as primary key to select or data to insert).
- Write code to submit SQL or stored procedures to Adaptive Server (for example, using dbsqlexec() calls).
- Write code to process result sets

- Name each transaction explicitly (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, use performance monitoring tools, available from third parties, to capture data streams.

Drivers

You need drivers for:

- General error handling
- Deadlock handling
- Result handling
- Time measurement
- Runtime 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

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. Sending output 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 multitiered applications where a bottleneck can occur at any level. Generally, 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 re-create a particular business situation.

Using a separate file for each transaction can prevent skewing; however, it requires additional code to synchronize 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.

Summary of tests

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

Table 6-2: Testing descriptions

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 expected functionality? • What are the limitations of the new features?	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 code errors related to multiuser 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
Integration testing	Ensures that all system components work well together, such as: • Batch processing • Online transaction processing (OLTP) • Decision support systems (DSS) and ad hoc queries • Operations, including backup, recovery, and dbcc commands • Sybase products other than Adaptive Server • Third-party products	Test suite models all system components

Stage	Purpose	Best technique
End-user acceptance testing	Executes acceptance tests specific to the environment. Also covers 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

Determining query processing changes

This section highlights Adaptive Server 15.0 query processing changes and features that may be useful during migration

See the *Adaptive Server Query Processor* guide for more information.

Before you start testing

Before you begin to diagnose query problems, make sure you:

- Installed the monitoring (MDA) tables and you have access to them.
- Enabled query monitoring in both the Adaptive Server 12.5 and 15.0 systems
- Have permission to run sp_configure, if needed.
- Have permission to turn on set command options in the query processor to get the diagnostics output.
- Are able to turn on trace flag 3604 and 3605
- Plan for sufficient file space; some of the outputs can be very large and may overwhelm your current space configuration.

- Practice capturing query plans using abstract query plans and practice running monitor tables to better understand how the tables work and how large you need to configure the pipes for statement, SQL text, and plan text.
- Create a test database to use as a practice area. You may bulk-copy output from the old and new servers into this database to perform SQL analysis, so create your test database on the 15.0 server to facilitate data copying.
- Plenty of free space (more than 2GB) for the system segment.

Determining which queries are impacted during migration

This section describes how to determine which queries are impacted during migration for Adaptive Server versions 12.5 and later to version 15.0

There are many ways you can identify queries impacted by changes in the query processing engine during migration. Each of the sections below describes a different scenario and a resolution.

You can automate this process with the Migration Analyzer in DBExpert 15.0, which compares the query plans and execution statistics between queries in Adaptive Server 12.5 and 15.0, identifying which queries have been changed and what the impact was.

Finding impacted queries

This section describes two methods for finding queries impacted by new query optimization techniques in Adaptive Server 15.0.

Capturing abstract query plans

Abstract query plans capture, load, and reuse query plans for executing queries. By default, captured query plans are stored in the sysqueryplans table. Both Adaptive Server 12.5.2 and 15.0 use query hash keys, so the query plans for identical queries can be matched.

This is an overview of the steps for capturing query plans:

1. Enable query plan capture on the 12.5 server. Do this at the session level with set plan dump on, or at the server level with sp_configure "abstract plan dump", 1.
2. On the 12.5 server, execute one module of the application you want to test.
3. Disable abstract query plan dump and bcp out sysqueryplans.

4 Enable query plan capture on the 15.0 server (use the same process as in step 1).

5 On the 15.0 server, execute the same module you executed in step 2.

6 Disable abstract query plan dump on the 15.0 server.

7 In the 15.0 server, create a table named queryplans_125.

8 bcp in the data from the 12.5 server into queryplans_125.

9 On the 12.5 server, copy the 15.0 data into the test database on the 12.5 server. Use select into to create a table called queryplans_150.

10 Create an index on hashkey, type, and sequence for both tables.

11 Run some of the sample queries listed below on both servers to identify plan differences. You can add to the complexity of these queries and eliminate the duplicates.

This generates a list of the queries that have changed:

```
select t.hashkey
   into #qpchgs
   from queryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence = f.sequence
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
      and t.text != f.text
union all
select f.hashkey
   from queryplans_150 f
   where f.sequence not in (select t.sequence
           from queryplans_125 t
           where f.hashkey = t.hashkey)
union all
select t.hashkey
from queryplans_125 t
where t.sequence not in (select f.sequence
                   from queryplans_150 f
                     where f.hashkey = t.hashkey)
go
```

This example eliminates duplicates:

```
select distinct hashkey
   into #qpchanges
   from #qpchgs
go
```

```
drop table #qpchgs
go
```

This example captures the SQL text for the queries identified:

```
select t.hashkey, t.sequence, t.text
from queryplans_125 t, #qpchanges q
where q.hashkey=t.hashkey
and t.type = 10-- sql text vs. aqp
```

To compare the abstract query plan text, first determine which 15.0 query plans may be longer:

```
select t.hashkey, t.sequence, t.text, f.sequence, f.text
   from ueryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence=*f.sequence
      and t.hashkey in (select hashkey from #qpchanges)
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
union all
```

Find the query that has a longer query plan in the 12.5 server (this may cause duplicates with the result set for the example above where the query plans for the 12.5. and the 15.0 server are equal):

```
select t.hashkey, t.sequence, t.text, f.sequence, f.text
   from ueryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence*=f.sequence
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
order by t.hashkey, t.sequence, f.sequence
```

Often an application executes the same query more than once. If this happens, and the query plans differ between 12.5 and 15.0 servers, the query results in duplicates in the result set of the last example.

Note None of these queries capture the execution metrics, so you cannot tell by looking at the output whether the query plans change the performance characteristics.

Using monitoring tables

This method for finding queries impacted by new query optimization techniques does not uniquely identify a query within the monitoring tables using the hashkey. This method relies on the monSysStatement monitoring table and related pipes, and uses only the monSysStatement, monSysSQLText, and monSysPlanText monitoring tables, and not the monProcessStatement, monProcessSQLText, and monProcessPlanText monitoring tables. Monitor tables that start with monSys... are "stateful" and keep track of historically executed statements, but tables that start with monProcess record only the currently executing statement.

Although the context of your connection's pipe state is retained for the current connection, it is independent of other connections (allowing multiuser access to the monitoring table historical data). For example, if you query the monSysStatement table and get a result of 100 rows, the next time you query this table, you see only the new rows added since the last query finished. However, if another DBA connects at this point and queries monSysStatement, his or her result set contains all the rows.

If you sample the monitor tables repeatedly using a polling process, you are viewing only the change in the table since your last query. If you reconnect, the session appears as a new session and the original state is lost, so the result set may contain rows already provided in the previous session.

Size the pipes correctly for your configuration. If the pipes are too small, statements or queries may be dropped from the ring buffer, and you must increase the number of available pipes or run sample queries more frequently. For example, if a module of an application submits 100,000 statements, setting the pipes to 100,000 may be impossible because of memory constraints. However, if you know these statements are issued over the period of an hour, then an average of 1,667 queries per minute are issued. If you guess at a peak of double, then 3,333 queries per minute are issued. In this situation, it is useful to set the pipes to 5,000 and sample every minute to avoid losing statements.

These are the general steps for finding queries impacted by new query optimization techniques using monitoring tables:

1 Configure the pipe setting in the 12.5 server for statement pipe active, sql text pipe active, and statement plan text pipe.

2 Issue a query similar to the following to create a temporary repository in tempdb for the monSysStatment, monSysSQLtext, and monSysPlanText tables:

```
select * into tempdb..monSysStatement
from master..monSysStatement where 1=2
```

3 Create a monitoring process that queries the monSysStatment, monSysSQLtext, and monSysPlanText tables once every minute and inserts the result set into the temporary tables you created in step 2. For example:

```
insert into tempdb..monSysStatement (select * from
master..monSysStatement)
```

To run this once every minute, you can place the query in a loop with `waitfor delay "00:01:00"`.

4 Execute one module of the application to be tested.

5 Stop the application and halt the monitoring.

6 bcp out the monitoring table you collected in step 3.

7 Repeat steps 1 – 5 on Adaptive Server 15.0.

8 Create a set of tables in the test database named after the monitoring tables but include version information in their names. For example, monSysStmt125, monSysStmt150, and so on.

9 Load the tables you created in step 8 with the information you collected in step 3 (use either bcp or an insert...select statement).

10 Create an index on the monSysStmt125 and monSysStmt15 tables on the SPID, KPID, DBID, ProcedureID, BatchID, ContextID, and LineNumber columns.

You can now search for queries that are impacted by the 15.0 server optimization techniques using the monSysStmt125 and monSysStmt15 tables.

Consider the following while you investigate queries:

- SQL queries (as opposed to triggers or procedures) have a ProcedureID value of 0.
- You can uniquely identify queries for a user with the SQL batch, which are counted in order by BatchID. The first SQL batch sent by an SPID is given a BatchID of 1, the second BatchID of 2, and so on.
- Within each SQL batch, the context (or “nesting level”) may change as triggers, procedures, or subprocedures are executed. The context may increment as the nesting increases and decrement as the execution drops from the nested procedure.
- Line numbers refer to the statement within the current context.

- This technique is accurate only within 100 milliseconds because it is based on the Adaptive Server CPU tick length, which defaults to 100 milliseconds. Statements that execute in less time than 100 milliseconds display as 0. This can cause a reporting error if, for example, an insert that took 10 milliseconds now takes 20 milliseconds, particularly if this insert is executed 1,000,000 times a day.

If you issue the same sequence of test statements, monSysStatements may be identical in both servers with the exception of the starting BatchID. You can use the SPID and KPID combinations on both server to identify the queries (assuming the application uses multiple connections).

For example, this query selects a list of SQL statements that executed more slowly on the 15.0 server than the 12.5 server. On the 12.5 server, the statements have SPIDs of 123, KPIDs of 4567890, and their BatchIDs started with 101; on the 15.0 server the statements have SPIDs of 24, KPIDs of 1234567, and their BatchID started with 12:

```
select f.BatchID, f.ContextID, f.LineNumber, CPU_15=f.CPUTime,
CPU_125=t.CPUTime,
     Wait_15=f.WaitTime, Wait_125=t.WaitTime,
     Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
     PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,

     LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
     Writes_15=f.PagesModified, Writes_125=t.PagesModified,
     ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
     ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
     DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
datediff(ms,t.StartTime,t.EndTime)
into #slow_qrys
from monSysStmt150 f, monSysStmt125
where f.SPID=24 and f.KPID=1234567
and t.SPID=123 and t.KPID=4567890
and t.BatchID=f.BatchID+(101-12)-- calculate offset for Batches.
and t.ContextID=f.ContextID
and t.LineNumber=f.LineNumber
and (datediff(ms,f.StartTime,f.EndTime) >
datediff(ms,t.StartTime,t.EndTime))
order by 18 desc, f.BatchID, f.ContextID, f.LineNumber
```

You can edit this same query to return the queries that run faster on a 15.0 server than a 12.5 server:

```
select f.BatchID, f.ContextID, f.LineNumber, CPU_15=f.CPUTime,
CPU_125=t.CPUTime,
     Wait_15=f.WaitTime, Wait_125=t.WaitTime,
     Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
     PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,
     LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
     Writes_15=f.PagesModified, Writes_125=t.PagesModified,
     ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
     ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
     DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
datediff(ms,t.StartTime,t.EndTime)
   into #slow_qrys
   from monSysStmt150 f, monSysStmt125
   where f.SPID=24 and f.KPID=1234567
     and t.SPID=123 and t.KPID=4567890
     and t.BatchID=f.BatchID+(101-12)-- calculate offset for Batches.
     and t.ContextID=f.ContextID
     and t.LineNumber=f.LineNumber
     and (datediff(ms,f.StartTime,f.EndTime) <
datediff(ms,t.StartTime,t.EndTime))
     order by 18 desc, f.BatchID, f.ContextID, f.LineNumber
```

Using sysquerymetrics

This method uses sysqueyrmetrics (which includes performance metric columns including logical IOs, CPU time, elapsed time, and so on) to find queries that are impacted by the 15.0 query processor. The method uses exact query matching, which is based on the hash key but yields execution statistics.

This method displays the query text but not the query plan, so you must first identify the slow queries in the 15.0 server and then compare the plans to those of the earlier server. The first part of this method is similar to the method that captures abstract query plans, but then uses sysquerymetrics to capture more information.

See the *Reference Manual: Tables* for a complete description of the sysquerymetrics table.

The steps are:

1. Enable query plan capture on the 12.5 server. You can do this at the session level with set plan dump on, or at the server level with sp_configure "abstract plan dump", 1.
2. Execute one module of the application to be tested on the 12.5 server.
3. Disable abstract query plans dump (set plan dump off) and bcp out sysqueryplans on the 12.5 server.
4. Enable query plan capture on the 15.0 server with set plan dump *group_name* on.

5 Enable metrics capture on the 15.0 server. You can do this at the server level with sp_configure "enable metrics capture", 1 or at the session level with set metrics_capture on.

6 Execute the same module from step 2 on the 15.0 server.

7 Disable abstract query plans dump (set plan dump off) on the 15.0 server.

8 On the 15.0 server, create a table named queryplans_125 in your test database; bulk-copy in the 12.5 data.

9 Create a table on the 15.0 server named queryplans_150 in your test database, and copy the 15.0 server abstract query plan data into this table.

10 Create an index on the hashkey, type and sequence columns for both abstract query plan tables.

11 Use sp_metrics to back up the sysquerymetrics data or copy it to table in the test database.

12 Run the following queries to identify plan differences.

This query creates a list of queries that have changed for the 15.0 server:

```
select t.hashkey
into #qpchgs
from queryplans_125 t, queryplans_150 f
where t.hashkey = f.hashkey
and t.sequence = f.sequence
and t.type = 100-- aqp text vs. sql
and f.type = 100-- aqp text vs. sql
and t.text != f.text
union all
select f.hashkey
from queryplans_150 f
where f.sequence not in (select t.sequence
             from queryplans_125 t
             where f.hashkey = t.hashkey)
union all
select t.hashkey
from queryplans_125 t
where t.sequence not in (select f.sequence
             from queryplans_150 f
             where f.hashkey = t.hashkey)
select distinct hashkey
into #qpchanges
from #qpchgs
go
drop table #qpchgs
```

This query selects a list of the queries that have changed for 15.0 and are running slowly. This query does not tell you if the queries ran faster or slower in the 15.0 server; it identifies queries in the 15.0 server that exceed a specified limit and whose query plans have changed from the 12.5 server release:

```
select hashkey, sequence, exec_min, exec_max, exec_avg,
     elap_min, elap_max, elap_avg, lio_min,
     lio_max, lio_avg, pio_min, pio_max, pio_avg,
     cnt, weight=cnt*exec_avg
     qtext
from <db>..sysquerymetrics-- database under test
where gid = <gid>-- group id sysquerymetrics backed up to
and elap_avg > 2000-- slow query is defined as avg elapsed time > 2000
and hashkey in (select hashkey from #qpchanges)
```

Finding long-running stored procedures

This section describes how to identify long-running stored procedures using sysquerymetrics to track statement-level statistics for each line and statement within a procedure, including subprocedures. This may be useful for capturing the parameters that may be driving the optimization within the procedure. This section describes how to use sysquerymetrics to identify long-running stored procedures.

This feature is new in Adaptive Server 15.0, so you cannot use the steps below to compare queries from previous releases. Often a query runs slowly because of a query plan that changed during migration.

Before you use this method, consider the following:

- The result set may skip or repeat LineNumbers because of looping or if ...else statements in the query, so subprocedures that are inside loops that are inside procedures may have the same ContextIDs, but different StartTimes.
- If you performed the upgrade in place using dump and load or mount database, the objectIDs of the user objects within the database are identical but the DBIDs differ.

- You must aggregate values to get procedure-level statistics:

Note The following is a single script, but has been divided into sections to describe the individual steps.

---

a Issue the following to aggregate to the individual procedure level for Adaptive Server 12.5:

```
select DBID, ProcedureID, StartTime, EndTime=max(EndTime),
   ElapsedTime=datediff(ms,StartTime,max(EndTime))/1000.00,
   CPUTime=sum(CPUTime), WaitTime=sum(WaitTime),
   LogicalReads=sum(LogicalReads),PhysicalReads=sum(PhysicalReads),
   PagesModified=sum(PagesModified)
into #procExecs125
from monSysStmt125
group by DBID, ProcedureID, Starttime
```

b Issue the following to aggregate to the individual procedure level for Adaptive Server 15.0:

```
select DBID, ProcedureID, StartTime, EndTime=max(EndTime),
   ElapsedTime=datediff(ms,StartTime,max(EndTime))/1000.00,
   CPUTime=sum(CPUTime), WaitTime=sum(WaitTime),
   LogicalReads=sum(LogicalReads), PhysicalReads=sum(PhysicalReads),
   PagesModified=sum(PagesModified)
into #procExecs150
from monSysStmt150
group by DBID, ProcedureID, Starttime
```

c Because both of these queries occur at individual procedure execution level, issue the following to average the result set if you run these queries more than once:

```
select DBID, ProcedureID, ExecCnt=count(*),
ElapsedTime=avg(ElapsedTime),
   CPUTime=avg(CPUTime), WaitTime=avg(WaitTime),
   LogicalReads=avg(LogicalReads), PhysicalReads=avg(PhysicalReads),
   PagesModified=avg(PagesModified)
into #procExecAvgs125
from procExecs125
group by DBID, ProcedureID
```

d To find the stored procedures that ran more slowly in the 15.0 server, issue:

```
select o.name as 'ProcName', o.type, ExecCnt=t.ExecCnt,
CPU_125=t.CPUTime, CPU_150=f.CPUTime,
   WaitTime_125=t.WaitTime, WaitTime_150=f.WaitTime,
   LogicalIO_125=t.LogicalReads, LogicalIO_150=f.LogicalReads,
   PhysReads_125=t.PhysicalReads, PhysReads_150=f.PhysicalReads,
   Writes_125=t.PagesModified, Writes_150=f.PagesModified,
   Elapsed_125=t.ElapsedTime, Elapsed_150=f.ElapsedTime
from #procExecAvgs125 t, #procExecAvgs150, <db>..sysobjects o
where t.ProcedureID=o.id
and t.ProcedureID=f.ProcedureID
and t.ElapsedTime < f.ElapsedTime
```

You can view the SQL text for the stored procedures that executed slowly by joining the result set from this query with monSysSQLText.

Because a parameter set may skew the average of one of the executions, you can join the first aggregated table (#procExecs) with monSysSQLText and view the respective execution metrics for each version.

e Issue this query to find the stored procedures that ran faster on the 15.0 server:

```
select o.name as 'ProcName', o.type, ExecCnt=t.ExecCnt,
   CPU_125=t.CPUTime, CPU_150=f.CPUTime,
   WaitTime_125=t.WaitTime, WaitTime_150=f.WaitTime,
   LogicalIO_125=t.LogicalReads, LogicalIO_150=f.LogicalReads,
   PhysReads_125=t.PhysicalReads, PhysReads_150=f.PhysicalReads,
   Writes_125=t.PagesModified, Writes_150=f.PagesModified,
   Elapsed_125=t.ElapsedTime, Elapsed_150=f.ElapsedTime
from #procExecs125 t, #procExecs150, <db>..sysobjects o
where t.ProcedureID=o.id
  and t.ProcedureID=f.ProcedureID
  and t.ElapsedTime > f.ElapsedTime
```

The steps for finding long-running queries using sysquerymetrics are:

1 Configure statement pipe active, sql text pipe active, and statement plan text pipe for the 12.5 server.

2 Create a temporary table in tempdb for monSysStatement, monSysSQLtext and monSysPlanText by issuing something similar to:

```
select * into tempdb..monSysStatement from
master..monSysStatement where 1=2
```

3 Create a monitoring process that queries the monSysStatment, monSysSQLtext, and monSysPlanText tables once every minute and inserts the result set into the temporary tables you created in step 2. For example:

```
insert into tempdb..monSysStatement (select * from
master..monSysStatement)
```

To run this once every minute, you can place the query in a loop with `waitfor delay "00:01:00"`.

4. On the 12.5 server, execute one module of the application to be tested.
5. Stop the application and halt the monitoring.
6. Bulk-copy out the monitoring table information you collected data from tempdb.
7. Repeat steps 1 – 5 for the 15.0 server.
8. Create a set of tables in the test database named after the monitoring tables but include version information in their names. For example: monSysStmt125, monSysStmt150, and so on.
9. Load the tables you created in step 8 with the information you collected in step 3 (use either bcp or insert...select statements).
10. Create an index on the monSysStThree3Ducksmt125 and monSysStmt15 tables on the SPID, KPID, DBID, ProcedureID, BatchID, ContextID, LineNumber columns.

Diagnosing and fixing query processing issues in 15.0 servers

This section discusses some tips for diagnosing problems with Adaptive Server 15.0.

Using sysquerymetrics and sp_metrics

Adaptive Server using sp_metrics to manage sysquerymetrics data. The data most currently collected is stored in sysquerymetrics with a global ID (GID) of 1. If you use the sp_metrics...backup parameter to save previous data, you must use a GID greater than 1 that is not already in use. To find the next available GID, select the `max(gid)` from sysquerymetrics and add one to the value.

sp_metrics examples

To enable metrics capture, enter:

```
set metrics_capture on
```

To flush the metrics, enter:

```
sp_metrics 'flush'
```

To select a GID (if this returns a NULL value or 1, use a value of 2 or higher), enter:

```
select max(gid)+1 from sysquerymetrics
```

To back up the data, enter:

```
sp_metrics 'backup', '3'
```

To turn off metrics capture, enter:

```
set metrics_capture off
```

To analyze the data, enter:

```
select * from sysquerymetrics where gid = 3
```

To drop the data, enter:

```
sp_metrics 'drop', '2', '5'
```

See the *Reference Manual: Commands* for more information about sp_metrics.

Filtering the data in sysquerymetrics

You can filter the data in sysquerymetrics by deleting data whose value is less than a predetermined value. You must enable allow updates before you delete data.

When you drop metrics, the begin range and end range must exist. For example, in this query, which attempts to drop the metrics from a group that starts with 3:

```
sp_metrics 'drop', '2', '5'
```

The drop fails because group 2 does not exist.

sysquerymetrics can consume a large amount of space in the system segment. However, to reduce the amount of space used:

1. Run sp_metrics capture.
2. At the end of the capture period, issue:

   ```
   Select max(gid) from sysquerymetrics
   sp_metrics 'backup', 'gid'
   ```

3. Start the next capture period.
4. Filter the previous results. For example, you can delete the rows that have an looming less than 10,000.
5. Repeat this sequence 10 to 15 times.

Using sysquerymetrics to support regression testing

You can use sysquerymetrics to support regression testing after changing the optimization goals, parallel resources, partitioning style for a table, or employing other Adaptive Server 15.0 features such as function-based indexes. The steps are:

- Enable sp_metrics capture for the initial run. These metrics are be used for reference.
- Run the application's module.
- Back up these reference metrics by setting their GID to 2.
- Make any configuration changes to improve the metrics
- Repeat steps 1 – 3, backing up to the next higher GID each time.

Identify queries that were impacted by joining sysquerymetrics on the hashkey. For example:

```
select r.hashkey, r.exec_avg, m.exec_avg, r.elap_avg, m.elap_avg,
     r.lio_avg, m.lio_avg, r.pio_avg, m.pio_avg, r.qtext
from sysquerymetrics r, sysquerymetrics m
where r.gid=2 and m.gid=<#>-- substitute # of current gid
and r.hashkey=m.hashkey
and ((r.exec_avg + (r.exec_avg * 0.1) < m.exec_avg)
     or (r.elap_avg + (r.elap_avg * 0.1) < m.elap_avg)
     or (r.lio_avg + (r.lio_avg * 0.1) < m.lio_avg)
     or (r.pio_avg + (r.pio _avg * 0.1) < m.pio_avg)
```

You can change the query to reflect the values you consider a regression, including adding a tolerance factor of 5. The example above includes a 10% tolerance factor to the reference times to avoid impacting the checkpoint process.

The example above has only one user, so it does not add `r.uid = m.uid`. You can change can be changed in the real application if multiple users executing the same module are to be tested).

Using showplan options

showplan in Adaptive Server 15.0 replaces many of the diagnostic trace flags used with set options. This is the syntax:

```
set option show_option (normal/brief/long/on/off)
```

See the *Reference Manual: Commands* for more information about set option syntax.

Usage issues

Before you use set option, consider the following:

- Some options require that you set dbcc traceon (3604) or 3605 to see the ouput on the client side or in the error log.
- Execute set option show on before you set other, more restrictive, options. This allows you to specify the brief or long options, which override the more general level of detail of the show command.
- If you enabled query metrics capture, set option may not produce output.

Example

This example displays the query plan, reports the statistics, and displays the abstract plan (which allows you to issue create plan to solve problems later on).

```
set showplan on
set option show_missing_stats on
set option show_abstract_plan on
```

To view missing statistics, issue:

```
        set option show_missing_stats long
        dbcc traceon(3604)
        go
        DBCC execution completed. If DBCC printed error
        messages, contact a user with System Administrator (SA)
        role.
select * from part, partsupp
where p_partkey = ps_partkey and p_itemtype = ps_itemtype
go
        NO STATS on column part.p_partkey
        NO STATS on column part.p_itemtype
        NO STATS on column partsupp.ps_itemtype
        NO STATS on density set for E={p_partkey, p_itemtype}
        NO STATS on density set for F={ps_partkey, ps_itemtype}
```

This example debugs the index selection, IO costing, and so on (previous releases did this with trace flags 302, 310, 315, and so on):

```
dbcc traceon(3604)
set showplan on
set option show long
```

set statistics plancost

The set statistics plancost option simplifies query analysis. It displays the estimated values for logical I/O, physical I/O, and row counts compared to the actual ones evaluated at each operator, and reports on CPU and sort buffer cost (`cpu: 0 bufct: 16` in the example below). To enable this display, enter:

```
dbcc traceon(3604)
go
set statistics plancost on
go
```

For example, this query:

```
select S.service_key, M.year, M.fiscal_period,count(*)
from telco_facts T,month M, service S
where T.month_key=M.month_key
  and T.service_key = S.service_key
  and S.call_waiting_flag='Y'
  and S.caller_id_flag='Y'
  and S.voice_mail_flag='Y'
group by M.year, M.fiscal_period, S.service_key
order by M.year, M.fiscal_period, S.service_key
```

Produces this query tree:

```
                                                  Emit
                                                   (VA = 7)
                                                   12 rows est: 1200
                                                    cpu: 500

                                           /
                                           GroupSorted
                                           (VA = 6)
                                            12 rows est: 1200

                                   /
                                   NestLoopJoin
                                   Inner Join
                                   (VA = 5)
                                   242704 rows est: 244857
                         /                   \
                        Sort                 IndexScan
                       (VA = 3)              month_svc_idx (T)
                       72 rows est: 24       (VA = 4)
                       lio: 6 est: 6         242704 rows est: 244857
                       pio: 0 est: 0         lio: 1116 est: 0
                       cpu: 0 bufct: 16      pio: 0 est: 0
              /
              NestLoopJoin
              Inner Join
              (VA = 2)

72 rows est: 24
/                       \
TableScan               TableScan
```

```
month (M)                service (S)
(VA = 0)                 (VA = 1)
24 rows est: 24          72 rows est: 24
lio: 1 est: 0            lio: 24 est: 0
pio: 0 est: 0             pio: 0 est: 0
```

If set statistics planview estimates the wrong row counts, the optimizer estimates are also off. This may be caused by missing or stale statistics. This is described in the following query (which is also run with show_missing_stats enabled):

```
dbcc traceon(3604)
go
set option show_missing_stats on
go
set statistics plancost on
go
select
            l_returnflag,
            l_linestatus,
            sum(l_quantity) as sum_qty,
            sum(l_extendedprice) as sum_base_price,
            sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
            sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
            avg(l_quantity) as avg_qty,
            avg(l_extendedprice) as avg_price,
            avg(l_discount) as avg_disc,
            count(*) as count_order
from
           lineitem
where
           l_shipdate <= dateadd(day, 79, '1998-12-01')
group by
           l_returnflag,
           l_linestatus
order by
           l_returnflag,
           l_linestatus
go
==================== Lava Operator Tree ====================


                                                    Emit
                                                    (VA = 4)
                                                    4 rows est: 100
                                                    cpu: 800
```

```
                                                /
                                                Restrict
                                                (0)(13)(0)(0)
                                                (VA = 3)
                                                4 rows est: 100

                                         /
                                         GroupSorted
                                         (VA = 2)

                                         4 rows est: 100
                                   /
                                   Sort
                                   (VA = 1)
                                   60175 rows est: 19858
                                   lio: 2470 est: 284
                                   pio: 2355 est: 558
                                   cpu: 1900 bufct: 21
                          /
                          TableScan
                          lineitem
                          (VA = 0)
                          60175 rows est: 19858
                          lio: 4157 est: 4157
                          pio: 1205 est: 4157

============================================================

NO STATS on column lineitem.l_shipdate

(4 rows affected)
```

The estimated number of rows is incorrect at the scan level. The query does have a predicate:

```
l_shipdate <= dateadd(day, 79, '1998-12-01')
```

If there are no statistics on `l_shipdate`, the query processor uses an arbitrary value. In this example, the query processor uses a value that creates an estimated row count of 19858 rows, which is far from the actual row count of 60175 rows, and is why the query processor decided to run a sort; the cost of a sort appears cheaper if the query processor estimates that the number of rows streamed into the sorter is a third of the actual count. The row reduction of the GroupSorted operator is also significant; reduced from 60175 to 4 rows, so the advantages of the GroupSorted algorithm are overridden by a hash-based grouping algorithm, which would probably cache all the data in memory.

Based on the value from the show_missing_stats option, the query processor decided to run update statistics on the l_shipdate column:

```
update statistics lineitem(l_shipdate)
```

If you re-run the query, this query tree is produced:

```
==================== Lava Operator Tree ====================

                                                      Emit
                                                      (VA = 4)
                                                      4 rows est: 100
                                                      cpu: 0
                                              /
                                              Restrict
                                              (0)(13)(0)(0)
                                              (VA = 3)
                                              4 rows est: 100
                                    /
                                    Sort
                                    (VA = 2)
                                    4 rows est: 100
                                    lio: 6 est: 6
                                    pio: 0 est: 0
                                    cpu: 800 bufct: 16
                          /
                          HashVectAgg
                          Count
                          (VA = 1)
                          4 rows est: 100
                          lio: 5 est: 5
                          pio: 0 est: 0
                          bufct: 16
                  /
                  TableScan
                  lineitem
                  (VA = 0)
                  60175 rows est: 60175
                  lio: 4157 est: 4157
                  pio: 1039 est: 4157
```

After update statistics runs, the estimated row count for the TableScan operator is the same as the actual row count, and the query plan changes to use the HashVectAgg (that is, hash-based vector aggregation) instead of the Sort and GroupSorted combination in the earlier example. The query also runs much faster.

You can perform more work to improve the query. The output of the HashVectAgg operator shows an estimated rowcount of 100, but the actual rowcount is 4. Because the grouping columns are on l_returnflag and l_linestatus, you can create a density on this pair of columns:

```
use tpcd
go
update statistics lineitem(l_returnflag, l_linestatus)
go
set showplan on
go
set statistics plancost on
go
```

If you re-run the query, you get this query plan:

```
QUERY PLAN FOR STATEMENT 1 (at line 2).

4 operator(s) under root

The type of query is SELECT.

ROOT:EMIT Operator

|RESTRICT Operator
|
|    |SORT Operator
|    | Using Worktable2 for internal storage.
|    |
|    |    |HASH VECTOR AGGREGATE Operator
|    |    |  GROUP BY
|    |    |  Evaluate Grouped COUNT AGGREGATE. |    |    |  Evaluate Grouped SUM OR
             AVERAGE AGGREGATE.
|    |    |  Evaluate Grouped COUNT AGGREGATE.
|    |    |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|    |    |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|    |    |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|    |    |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|    |    |  Using Worktable1 for internal storage.
|    |    |
|    |    |  |  SCAN Operator
|    |    |  |  FROM TABLE
|    |    |  |  lineitem
|    |    |  |  Table Scan.
|    |    |  |  Forward Scan.
|    |    |  |  Positioning at start of table.
|    |    |  |  Using I/O Size 2 Kbytes for data pages.
|   |   |  |  With MRU Buffer Replacement Strategy for data pages.

==================== Lava Operator Tree ====================
                                                    Emit
                                                   (VA = 4)
                                                   4 rows est: 4
                                                   cpu: 0

                                          /
                                          Restrict
                                          (0)(13)(0)(0)
                                          (VA = 3)
                                          4 rows est: 4

                                  /
                                  Sort
                                  (VA = 2)
                                  4 rows est: 4
                                  lio: 6 est: 6
                                  pio: 0 est: 0
                                  cpu: 700 bufct: 16
                         /
                         HashVectAgg
                         Count (VA = 1)
                         4 rows est: 4
                         lio: 5 est: 5
                         pio: 0 est: 0
                         bufct: 16
                  /
                  TableScan
                  lineitem
                  (VA = 0)
                  60175 rows est: 60175
                  lio: 4157 est: 4157
                  pio: 1264 est: 4157
```

The estimated row count for HashVectAgg is same as that of the actual row count.

Query plans as XML

Adaptive Server 15.0 allows you to view query plans in XML, which you can use to build automated tools, such as the Plan Viewer in the Adaptive Server plug-in to display the query plan graphically. You can also use an XML query plan to find the last time statistics were updated for a particular table. Earlier Adaptive Server releases did this using the optdiag utility or by querying systabstats or sysstatistics.

Viewing query plans with XML is a single-step operation. As a textual representation of the query, showplan does not provide statistical information, and you need to perform an additional step to determine the last time statistics were updated by using optdiag or querying the system tables. All statistical information is available in a single location with the XML output, and because parsing XML is much simpler than parsing text, tool developers can more easily create enhanced functionality for Adaptive Server 15.0 than earlier releases.

Query-level debugging in Adaptive Server 15.0

This section describes debugging techniques for Adaptive Server release 15.0.

Using optdiag for statistics

You can use optdiag to determine if the current statistics are stale and you need to generate new statistics. This example uses optdiag to capture the statistics for the part table of the le_01 database:

```
$SYBASE/ASE-15_0/bin/optdiag statistics le_01.dbo.part -Usa -P

Server name: "tpcd"

Specified database: "le_01"
Specified table owner: "dbo"
Specified table: "part"
Specified column: not specified

Table owner: "dbo"
Table name: "part"

...................................................
Statistics for column: "p_partkey"
Last update of column statistics: Sep 13 2005 7:51:39:440PM

Range cell density: 0.0010010010010010
Total density: 0.0010010010010010
Range selectivity: default used (0.33)
```

```
In between selectivity: default used (0.25)

Histogram for column: "p_partkey"
Column datatype: integer
Requested step count: 20
Actual step count: 20
Sampling Percent: 0

Step Weight Value

1 0.00000000 <= 0
2 0.05205205 <= 52

.......................................................
Statistics for column: "p_brand"
Last update of column statistics: Sep 13 2005 7:51:39:440PM

Range cell density: 0.0010010010010010
Total density: 0.0010010010010010
Range selectivity: default used (0.33)
In between selectivity: default used (0.25)
```

Using show_final_plan_xml for statistics

With Adaptive Server 15.0 and later, you can use XML to determine when statistics were last updated. Specify a query on which you want to run statistics, and Adaptive Server displays only those statistics that are useful for the query, including when the statistics were last updated for these columns. optdiag shows statistics for all indices whether or not they are used, and requires multiple executions if more than one table is involved. The following collects statistics for the part table from the previous example:

```
1> set plan for show_final_plan_xml to message on
2> go
1> select count(*) from part where p_partkey > 20
2> go

-----------

979
1> select showplan_in_xml(-1)
2> go


-----------
979
<?xml version="1.0" encoding="UTF-8"?>
<query>
             <planVersion> 1.0 </planVersion>
```

```
                <statementNum>1</statementNum>
                <lineNum>1</lineNum>
                <text>
                      <![CDATA[
                         SQL Text: select count(*) from part where p_partkey >
20
                      ]]>
                   </text>
                      <objName>part</objName>
                      <columnStats>
                         <column>p_partkey</column>
                         <updateTime>Sep 13 2005 7:51:39:440PM</updateTime>
                      </columnStats>
```

Sending XML data directly to the client

This example uses trace flag 3604 and the client parameter to send the information from show_final_plan_xml to the client:

```
1> dbcc traceon(3604)
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
set plan for show_final_plan_xml to client on
go
select * from part, partsupp
where p_partkey = ps_partkey and p_itemtype = ps_itemtype
go
<?xml version="1.0" encoding="UTF-8"?>
<query>
<planVersion> 1.0 </planVersion>
<optimizerStatistics>
<statInfo>
                     <objName>part</objName>
                     <missingHistogram>
                             <column>p_partkey</column>
                             <column>p_itemtype</column>
                     </missingHistogram>
                     <missingDensity>
                             <column>p_partkey</column>
                             <column>p_itemtype</column>
                     </missingDensity>}
               </statInfo>
               <statInfo>
                     <objName>partsupp</objName>
                     <missingHistogram>
                             <column>ps_partkey</column>
                             <column>ps_itemtype</column>
                     </missingHistogram>
                     <missingDensity>
```

```
                        <column>ps_partkey</column>
                        <column>ps_itemtype</column>
                </missingDensity>
            </statInfo>
        </optimizerStatistics>
```

Early timeout detection versus tablecount

The 15.0 query processor may automatically activate a timeout mechanism to reduce the time spent in the search engine. You can also configure a more aggressive timeout to reduce the amount of procedure cache used by the query processor. The valid values are from 0 to 1000.

You cannot configure Adaptive Server to have no timeout period.

A timeout is common when you have a query with a large number of tables (that is, greater than 3). The disadvantage of a timeout is that it may miss the best plan because it exits early from the optimizer. However, an advantage of a timeout is that it reduces the time spent optimizing the query when query optimization might exceed execution time. For example, in a test using a 64-bit 11.9.3 Adaptive Server, a 12-way join with tablecount set to 12 took 10 minutes to optimize but only 30 seconds to execute. This was was made worse because the optimizer found the optimal plan within the first minute, but spent most of the time searching the 12 factorial (479 million) possible combinations.

Use the set option show on parameter to detect when timeouts occur. If a timeout occurs, the following appears in the diagnostics output:

```
!! Optimizer has timed out in this opt block !!
```

Raising the value of the timeout period at the server or session level can hurt other queries, and may consume more resources such as procedure cache, and may increase your compilation time. In general, if you know that a specific stored procedure or query is creating a problem, you may want to force recompilation of it with a higher value for the optimization timeout.

The syntax for setting the timeout period at the server level is:

```
sp_configure "optimization timeout limit",
time_out_period
```

The syntax for setting the timeout period at the session level is:

```
set plan opttimeoutlimit time_out_period
```

At the query level, you can force the timeout by issuing:

```
select * from table_name plan "(use opttimeoutlimit
time_out_period)"
```

Fixing queries using abstract query plans

Abstract query plans allow customers to modify queries to use a specific plan without altering the query code. The steps for using abstract query plans, which are transparent to applications, are:

1. Identify the queries that have problems in the 15.0 server.
2. Enable abstract query plans to capture query plans in the earlier version of the server and run the queries that pose problem with the 15.0 server.
3. Extract the abstract query plans from the ap_stdout group.
4. Review and modify the abstract queries to adjust for partial plan use or other issues that may be affecting the query's performance.
5. In the 15.0 server, use create plan to load the query plans into the default ap_stdin group in the 15.0 server.
6. Enable abstract plan load on the 15.0 server.
7. Disable abstract plan cache on the 15.0 server to avoid IO during optimization.
8. Re-run the queries in the 15.0 server to determine if the abstract query plans are used.
9. As Adaptive Server introduces new features that improve query performance, you may need to adjust the abstract query plans.
10. Work with Sybase Technical Support to resolve any original optimization issue.

For more information about abstract query plans, see the *Performance and Tuning Guide:Optimizer and Abstract Plans*.

These steps are useful to resolve problems that delay an upgrade beyond the window of opportunity for diagnosing the problem with the query, rewriting the query, and resolving any issues that occur. You can use abstract plans to replace set forceplan on, which allows you more control than the join-order processing enforcement that forceplan implements, which is illustrated in the following examples.

Forcing an index

This example scans the lineitem table without an index.

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
QUERY PLAN FOR STATEMENT 1 (at line 1).
3 operator(s) under root
The type of query is SELECT.

ROOT:EMIT Operator

|NESTED LOOP JOIN Operator (Join Type: Inner Join)
|
|   |SCAN Operator
|   |  FROM TABLE
|   |  orders
|   |  Table Scan.
|   |  Forward Scan.
|   |  Positioning at start of table.
|   |SCAN Operator
|   |  FROM TABLE
|   |  lineitem
|   |  Table Scan.
|   |  Forward Scan.
|   |  Positioning at start of table.
```

This method may not create the best available query plan, and may run faster if you use the l_idx1 index on lineitem. Try rewriting the query to force the index:

```
select count(*) from orders, lineitem (index l_idx1) where o_orderkey =
l_orderkey
QUERY PLAN FOR STATEMENT 1 (at line 1).
3 operator(s) under root
The type of query is SELECT.

ROOT:EMIT Operator

|NESTED LOOP JOIN Operator (Join Type: Inner Join)
|
|   |SCAN Operator
|   |  FROM TABLE
|   |  orders
|   |  Table Scan.
|   |  Forward Scan.
|   |  Positioning at start of table.
|
|   |SCAN Operator
|   |  FROM TABLE
|   |  lineitems
|   |  Index : l_idx1
```

```
    |   |  Forward Scan.
    |   |  Positioning by key.
    |   |  Keys are:
    |   |  l_orderkey ASC
```

Forcing an index with abstract plans

Although using the force parameter often solves query plan issues, it requires that you change the application code. Even if changing the code is not a problem for you, this can take much longer than using an abstract query plans.

The following examples run the same query as above, but uses abstract plans to improve them.

First, enable abstract plans:

```
set option show_abstract_plan on
go
dbcc traceon(3604)
go
```

Adaptive Server generates the abstract query plans, which you edit and then force to use an index.

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
go
The Abstract Plan (AP) of the final query execution plan:
( nl_join ( t_scan orders ) ( t_scan lineitem ) ) ( prop orders ( parallel
1 ) ( prefetch 2 ) (lru ) ) ( prop lineitem ( parallel 1 ) ( prefetch 2 ) (
lru ) )
```

You can modify the abstract plan to change the query processor's behavior, and pass the query plan to the query processor using the PLAN clause. The syntax is:

```
SELECT/INSERT/DELETE/UPDATE ...PLAN '( ... )
```

To improve the plan, replace the table scans (`t_scan`) with index accesses and specify the tables using `(prop` ***`table_name`***`)` (the example is indented for display purposes):

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
plan
"( nl_join
            ( t_scan orders )
            ( t_scan lineitem )
           )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

To force the index scan, use (i_scan *index_name table_name*) parameter:

```
1> select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan orders )
         ( i_scan l_idx1 lineitem )
     )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

Forcing an join order

In previous releases, you forced join orders by using set forceplan on.

This example forces the join order so the lineitem table is outer to the orders table, so their join orders are switched in the from clause of the query:

```
set forceplan on
2> go
1> select count(*) from lineitem, orders where o_orderkey = l_orderkey
2> go
```

You can also force the join order with abstract plans. After you enable abstract plans, the query looks like this:

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan orders )
         ( t_scan lineitem )
     )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

You can force the join order with abstract plans by switching their join order:

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan lineitem)
         ( t_scan orders )
     )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

Forcing different subquery attachments

You can change a subquery attachment only for correlated subqueries that cannot be flattened. Changing the subquery attachment reduces the number of times a subquery gets evaluated.

For example, in the three-table join shown below, only the skeletal plan output shows that the subquery is attached after the join of the three outer tables is performed. This is highlighted in the showplan output.

```
1> select count(*)
2> from lineitem, part PO, customer
3> where l_partkey = p_partkey and l_custkey = c_custkey
4> and p_cost = (select min(PI.p_cost) from part PI where PO.p_partkey =
PI.p_partkey)
5> go

The Abstract Plan (AP) of the final query execution plan:
( scalar_agg ( nested ( m_join ( sort ( m_join ( sort ( t_scan customer ) ) (
sort (t_scan lineitem ) ) ) ) ( i_scan part_indx (table (PO part ) ) ) ( subq
( scalar_agg ( t_scan (table (PI part ) ) ) ) ) ) ) (prop customer ( parallel
1 ) ( prefetch 2 ) ( lru ) ) ( prop (table (PO part)) ( parallel 1 ) (prefetch
2 ) ( lru ) ) (prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) ) ( prop
(table(PI part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
```

You can modify the abstract plan to change the query processor’s behavior, and pass the query plan to the query processor using the PLAN clause. The syntax is:

```
SELECT/INSERT/DELETE/UPDATE ...PLAN '( ... )
```

The new plan is:

```
QUERY PLAN FOR STATEMENT 1 (at line 1).

12 operator(s) under root

The type of query is SELECT.

ROOT:EMIT Operator

|SCALAR AGGREGATE Operator
| Evaluate Ungrouped COUNT AGGREGATE.
|
|   |SQFILTER Operator has 2 children.
|   |   |MERGE JOIN Operator (Join Type: Inner Join)
|   |   |
|   |   |   |SORT Operator
|   |   |   | Using Worktable4 for internal storage.
|   |   |   |
|   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join)
|   |   |   |   |
|   |   |   |   |   |SORT Operator
|   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |
|   |   |   |   |   |   |SCAN Operator
|   |   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |   |  customer
|   |   |   |   |   |   |  Table Scan.
|   |   |   |   |   |  SORT Operator
|   |   |   |   |   | Using Worktable2 for internal storage.
|   |   |   |   |   |
|   |   |   |   |   |   |SCAN Operator
|   |   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |   |  lineitem
|   |   |   |   |   |   |  Table Scan.
|   |   |   |SCAN Operator
|   |   |   |  FROM TABLE
|   |   |   |  part
|   |
|   |   Run subquery 1 (at nesting level 1).
|   |
|   |   QUERY PLAN FOR SUBQUERY 1 (at nesting level 1 and at line 4).
|   |
|   |   Correlated Subquery.
|   |   Subquery under an EXPRESSION predicate.
|   |
|   |   |SCALAR AGGREGATE Operator
|   |   |  Evaluate Ungrouped MINIMUM AGGREGATE.
| | |
|   |   |   |SCAN Operator
|   |   |   |  FROM TABLE
|   |   |   |  part
|   |
|   |   END OF QUERY PLAN FOR SUBQUERY 1.
```

You can view the operator tree with trace flag 526 or set statistics plancost. Trace flag 526 displays the operator tree without the cost (in the example below). However, knowing the cost is useful for determining whether the abstract plan you are forcing is efficient, so Sybase recommends that you use set statistics plancost on . In this example, note the location of the aggregate on PI.

```
==================== Lava Operator Tree ====================
                       Emit
                       (VA = 12)

                       /
                    ScalarAgg
                    Count
                    (VA = 11)
                    /
                  SQFilter
                  (VA = 10)

              /           \
            MergeJoin       ScalarAgg
            Inner Join        Min
            (VA = 7)          (VA = 9)
            /       \             /
         Sort        TableScan   TableScan
         (VA = 5)     part(PO)   part(PI)
                     (VA = 6)    (VA = 8)
         /
      MergeJoin
      Inner Join
      (VA = 4)
      /        \
Sort            Sort
(VA = 1)        (VA = 3)

/                   /
TableScan          TableScan
customer           lineitem
(VA = 0)           (VA = 2)
```

This query plan may not be optimal. The subquery is dependent on the outer table `part (PO)`, and can be attached anywhere after this table has been scanned.

This example assumes that the correct join order must be part (PO) as the outermost table, followed by lineitem and then customer as the innermost table. If you need to attach the subquery to the scan of table PO, start with the abstract plan produced in the previous example and then modify the query as necessary:

```
select count(*)
from lineitem, part PO, customer
where l_partkey = p_partkey and l_custkey = c_custkey
and p_cost = (select min(PI.p_cost) from part PI where PO.p_partkey =
PI.p_partkey)
plan
```

```
"(scalar_agg
        (m_join
          (sort
             (m_join
                   (nested
                      (scan (table (PO part)))
                      (subq (scalar_agg (scan (table (PI part)))))
                    )
                   (sort
                      (scan lineitem)
                   )
              )
         )
         (sort
            (scan customer)
         )
       )
    )
(prop customer ( parallel 1 ) ( prefetch 2 ) ( lru ) )
(prop (table (PO part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
(prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
(prop (table(PI part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
go
==================== Lava Operator Tree ====================

                            Emit
                            (VA = 12)
                            /
                            ScalarAgg
                            Count
                            (VA = 11)
                          /
                         MergeJoin
                         Inner Join
                         (VA = 10)
                         /        \
                      Sort        Sort
                      (VA = 7)    (VA = 9)

                     /                  /
               MergeJoin               TableScan
               Inner Join              customer
               (VA = 6)                (VA = 8)
               /         \
           SQFilter        Sort
           (VA = 3)        (VA = 5)
           /      \            /
       TableScan   ScalarAgg  TableScan
       part(PO)    Min        lineitem
       (VA = 0)     (VA = 2)   (VA = 4)
                      /
                  TableScan
                  part (PI)
                  (VA = 1)

============================================================
The type of query is SELECT.

ROOT:EMIT Operator

|SCALAR AGGREGATE Operator
| Evaluate Ungrouped COUNT AGGREGATE.

|   |MERGE JOIN Operator (Join Type: Inner Join)
|   |  Using Worktable5 for internal storage.
|   |  Key Count: 1
|   |  Key Ordering: ASC
|   |
|   |   |SORT Operator
|   |   |  Using Worktable3 for internal storage.
|   |   |
|   |   |   |MERGE JOIN Operator (Join Type: Inner Join)
|   |   |   |  Using Worktable2 for internal storage.
|   |   |   |  Key Count: 1
|   |   |   |  Key Ordering: ASC
|   |   |   |
|   |   |   |   |SQFILTER Operator has 2 children.
|   |   |   |   |
|   |   |   |   |   |SCAN Operator
|   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |  part
|   |   |   |   |   |  PO
|   |   |   |   |   |  Table Scan.
|   |   |   |   |   |  Forward Scan.
|   |   |   |   |
|   |   |   |   | Run subquery 1 (at nesting level 1).
|   |   |   |   |
|   |   |   |   | QUERY PLAN FOR SUBQUERY 1 (at nesting level 1 and at line 4).
|   |   |   |   |
|   |   |   |   | Correlated Subquery.
|   |   |   |   | Subquery under an EXPRESSION predicate.
|   |   |   |   |
|   |   |   |   |   |SCALAR AGGREGATE Operator
|   |   |   |   |   |  Evaluate Ungrouped MINIMUM AGGREGATE.
|   |   |   |   |   |
|   |   |   |   |   |SCAN Operator
|   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |  part
|   |   |   |   |   |  PI
|   |   |   |   |   |  Table Scan.
|   |   |   |   |   |  Forward Scan.
|   |   |   |   |  END OF QUERY PLAN FOR SUBQUERY 1.
|   |   |   |
|   |   |   |   |SORT Operator
|   |   |   |   |  Using Worktable1 for internal storage.
|   |   |   |   |
|   |   |   |   |   |SCAN Operator
|   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |  lineitem
|   |   |   |   |   |  Table Scan.
|   |   |   |   |   |  Forward Scan.
|   |   |SORT Operator
|   |   |  Using Worktable4 for internal storage.
|   |   |
|   |   |   |SCAN Operator
|   |   |   |  FROM TABLE
|   |   |   |  customer
|   |   |   |  Table Scan.
|   |   |   |  Forward Scan.
```

This example shifts ScalarAgg and TableScan compared to its placement in the subquery in the previous example.

Reporting query processor and optimizer issues

There are several types of query processing problems, but they typically produce a stack trace in the error log (possibly coupled with a client disconnect) or degraded performance (either from an unknown cause, or because you cannot force a correct query plan). When a problem occurs, you must isolate the problem query. After this, collect the output listed below and contact Sybase Technical Support. Sybase has non-disclosure agreements with all customers, which may alleviate your concerns about providing business-sensitive data.

- A full database dump – the preferred output. However, if a full database dump is not available, provide the full schema of the tables involved, stored procedure source code, and a bcp extraction of the data. Although you can resolve some issues without this information, the information you are providing is made up of approximations, and does not guarantee resolution.. By having a copy of the data, the exact data cardinality, and the data volumes available, you also have the methods the query processor uses to select data for costing algorithms, not only for problem detection, but also for testing the resolution.
- ddlgen output – try to provide the full schema for all tables, indices, triggers, and procedures involved.
- optdiag output (including the simulate mode, if used) – Sybase must have some notion of your data volumes, cardinalities, and how the query processor selects the data. If you influence the query processor using simulated statistics, include those as well.
- Forced plan information – force the plan that runs best and collect the following information on it, as well as the same information about the plan that does not run well:
  - set statistics plancost on.
  - set statistics time on.
  - set option show long. The output from this can be very large. You many need trace flag 3604 enabled to run this.
  - set showplan on (with trace flag 526 enabled).

If you are receiving incorrect results from queries, what you provide Sybase depends on whether you are running with or without parallelism:

- Without parallelism – whatever you can provide in terms of database dumps or bcp extracts of the data is most helpful. However, in addition, collect the output after enabling the following options and running the query:
  - set option show_code_gen on
  - dbcc traceon(201)
- With parallelism – If your problem occurs only in parallel queries but does not occur when the server is run in serial, in addition to the ones listed above, include the following:
  - set option show long

- set option show_parallel long
- set option show_elimination long (if you do not use proper partition elimination)

Testing for performance

This section expands on performance benchmarking before and after the upgrade process including:

- Pre-upgrade single-user tests
- Pre-upgrade multiuser tests
- Test system upgrade
- Post-upgrade single-user tests
- Post-upgrade multiuser tests

Note Benchmarks test server processing speed. Therefore, run your benchmarks from a lightweight client so all processing takes place at the back end.

Pre-upgrade single-user tests

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

Optimizer

Compare the index selectivity and IO projections from the 300-series trace flags in the 12.x Adaptive Servers with show option output in the 15.x Adaptive Server. Some possible causes of optimizer differences include:

- Data and data distribution problems. Check data layout and verify that these match the production 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 continuing.

If you captured abstract plans in the previous release of Adaptive Server, use the query metrics feature to compare plans and performance on the new version. See the “Query Processing Metrics” chapter in the *Query Processor* manual at http://sybooks.sybase.com/as.html for details.

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 multiuser tests

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

Untimed benchmarks

In multiuser 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 you run timed benchmarks. Gather response time and throughput metrics. Run the tests as often as necessary to fix problems and reproduce the final results.

Test system upgrade

Perform the upgrade on the test system following the instructions in the installation guide. 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 5, “Making Database Administration Changes”. However, do not change your Adaptive Server configuration to use new features at this time.

Note Your first objective after upgrading 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 upgrading:

- Perform dbccs 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 equivalent or better query plans. Differences in query plan may be due to optimizer changes, depending on the Adaptive Server version you were running. Determine whether you need to change application code to compensate. See Chapter 6, “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 multiuser tests

After reloading backups to return the test system to a known state, rerun your benchmark testing.

Untimed benchmarks

In multiuser 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: Monitoring and Analyzing* for more information. See also the 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 you run 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 reproduce the final results.

APPENDIX A Worksheets for Your Current Environment

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

Topic	Page
Adaptive Server operational worksheets	141
Data architecture worksheet	145
Adaptive Server infrastructure worksheets	148

As part of migration planning, see the ASE Migration Resources Web page at http://sybase.com/support/techdocs/migration for current documents. You can learn more about SySAM in the *User's Guide Sybase Software Asset Management* and at http://www.sybase.com/sysam.

Adaptive Server operational worksheets

The following worksheets are for gathering business requirements. For more information, see Chapter 1, "Documenting Business Requirements."

This worksheet includes the following sections:

- Operational business requirements
- Backup and restore procedures
- Database dump details
- Maintenance procedure details

Operational business requirements

The following worksheet is for recording operational business 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.

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 off site?		
Are maintenance activities documented?		
What is the frequency of Adaptive 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:

Database Name	Frequency of database dumps	Dump device used	Frequency of transaction log dumps	Dump device used	Comments

Database Name	Frequency of database dumps	Dump device used	Frequency of transaction log dumps	Dump device used	Comments

Maintenance procedure details

Use the following worksheet to record detailed maintenance information.

Database/ object Name	Frequency of *dbcc checkdb/checktable*	Frequency of *dbcc checkalloc/tablealloc*	Frequency of *update statistics*	Frequency of monitoring space utilization

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 1, "Documenting Business Requirements.". The data architecture worksheet includes these sections:

- Client application components
- Production performance metrics
- Transaction profile

Client application components

Use the following worksheet to record application profile and performance requirements.

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

Table A-1 describes the current production performance metrics to measure using operating system monitors.

Table A-1: Metrics for production system

Metric	What to measure
CPU	The average and maximum CPU utilization (aggregate and per CPU on SMP servers) per “time window” (online, batch, and so on) per server.

Metric	What to measure
Disk I/O	• Number of I/Os per second per disk and controller, and I/O queue lengths per "time window" per server. • 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	• The packets per second per NIC per "time window" per server. • 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

Record transaction profile information for each database on the server.

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 timer required	Current average and maximum response time

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 timer required	Current average and maximum response time

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

Adaptive Server infrastructure worksheets

Documenting your environment is discussed in Chapter 3, "Writing a Migration Plan." The infrastructure worksheets include:

- Host configuration
- Adaptive Server configuration
- Database devices
- Databases and segments
- Dump devices

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:

Adaptive Server machine
Make:
Model:
Customer ID with hardware vendor:
Technical support phone number:
Technical support hours:
Technical account manager: • Name: • Phone numbers: • Pager numbers:

Record CPU resources:

CPU
Number of physical processors:
Chip speed:
Number of processors available to Adaptive Server:
Other CPU-intensive processes/threads:

CPU
Processes/threads bound to specific CPUs:
Processes/threads run at high priority:

Physical memory usage

List all the major processes and memory requirements running on each server.

Application	Runtime memory usage
Operating system	
Adaptive Server total memory + additional netmem + extent i/o buffers	
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 incompatibilities and capacity planning.

Controller number	Make and model	Hardware revision	Months in service	Transfer rate (KB/sec)

The following disk layout information can help with hardware incompatibilities, mean-time between failures (MTBF) that are approaching their limits, load balancing, and capacity planning.

Physical device name	Make and model	Hardware 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 if redistribution is required, for load balancing, and for capacity planning.

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 if redistribution is required, for load balancing, and for capacity planning.

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.

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.

Physical device name	Make and model	Firmware revision	Months in service	Capacity (MB)	Controller number	Transfer rate (KB/secon d)

Operating system configuration

Record details about your operating system.

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 numbers: • Pager number:

Adaptive Server configuration

Record general information about the Adaptive Server configuration.

General configuration
Home directory:
Page size:
Components, release and fix levels:

General configuration
Locations/names of scripts to rebuild database environment:
sp_configure configuration values:
buildmaster configuration values:
License_key information

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

Usage	MB	Page size	Bytes
Configured memory			
Code size			
Kernel structures			
Server structures			
Page cache			
Proc buffers			
Proc headers			

Usage	MB	Page size	Bytes
Number of page buffers			
Number of proc buffers			

Database devices

Database device information can help id redistribution is required, for load balancing, and capacity planning.

Database device name	Physical device name	Virtual device number	Capacity (pages)	Page size	Mirrored device name

Databases and segments

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

Database name	Database options set	Fragment (page range)	Size (in MB)	Segment names	Device name

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.

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

APPENDIX B Sample Migration Task Lists

This appendix provides the following samples.

Topic	Page
Sample task list template	159
General migration task list example	160
Parallel migration task list example	167
Cutover migration task list example	173
Staged cutover task overview	178

Note These examples are illustrations. They are not step-by-step procedures that will work as-is for every site. Every migration is unique and you must write a plan that is specific for your site.

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. Your own migration task lists may differ in detail and order. These general areas are included:

- Migration analysis
- Migration preparation
- Implement migration (using install/load technique)
- Implement migration (using upgrade technique)
- Migration quality assurance

See Chapter 3, “Writing a Migration Plan” for more information on migration planning.

Migration analysis

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 data server 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-platform 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. Verify new database conversion and computations.
3. Search for queries containing changed SQL syntax.
4. Design code changes.
5. 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 migration 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 premigration 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 preparation 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, and so on).
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 Ensure no access to obsolete files.
8 Notify users of availability.

Implement migration (using upgrade technique)

Upgrade adaptive server

1 Verify target system readiness.
2 Configure file system.
3 Install software.
4 Use sqlupgrade 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 Ensure no access to obsolete files.
8 Notify users of availability.

Migration quality assurance

Perform aystem 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 in to Adaptive Server production environment.
5. Back up new environment.
6. Notify users of data refresh.

Parallel migration task list example

This sample task list covers a parallel migration with replication approach upgrading to Adaptive Server Enterprise 15.0.

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 the company cannot afford more than 1 hour of data loss. The migration must support these contingency requirements if a production backout is required.

This example does not repeat the preparation steps to given in the previous example. The task lists for this example include:

- Define test/acceptance criteria—regression test suites
- Set up target production environment
- Set up Replication Server
- Run regression test suites
- Upgrade server B (shadow)
- Run post-upgrade regression test suites on Adaptive Server 15.0 (server B)
- Run user acceptance tests on Adaptive Server 15.0 (server B)
- Shift production users to Adaptive Server 15.0 (server B)
- Perform final steps

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 code for target user functions.
3. Encapsulate SQL code 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 older system on Server B.

5 Configure older system 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 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 ..., and so on).
4. Verify and document older system regression test.

Front-end simulation regression test suite

1. Iteratively perform front-end simulation regression tests.
2. Monitor and capture system dynamics (sp_who, sp_lock ..., and so on).
3. Verify and document older system 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 ..., and so on).
4. Decide whether the shadow server is ready to be upgraded.
5. Verify and document older system regression test.
6. Verify performance and functions on Server B.

Upgrade server B (shadow)

1. Alter sybsystemprocs.
2. Perform pre-upgrade verification.
3. Upgrade Server B to Adaptive Server version 15.0.
4. Load Adaptive Server 15.0 environment from media dumps.
5. Run dbcc commands (checktable, checkalloc, checkcatalog).
6. Set baseline 15.0 configuration parameter values.

7 Perform dump of older databases from Server A.

8 Load dump on to 15.0 system.

9 Run dbcc commands (checktable, checkalloc, checkcatalog) on Adaptive Server 15.0 databases. Verify the dbcc log and error log.

10 Recycle Adaptive Server.

Run post-upgrade regression test suites on Adaptive Server 15.0 (server B)

Back-end regression test suite—production loads

1 Perform back-end regression testing.

2 Monitor and capture system dynamics (sp_who, sp_lock, statistics io...).

3 Verify and document 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 regression test.

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 regression test.

Other testing

Verify 15.0 performance and functions.

Run user acceptance tests on Adaptive Server 15.0 (server B)

1. Have production testers perform user regression testing of version 15.0.
2. Monitor and capture system dynamics (sp_who, sp_lock...).
3. Verify and document regression test.
4. Determine if version 15.0 is successful.

Shift production users to Adaptive Server 15.0 (server B)

1. Ensure there is no production activity.
2. Perform a dump from Server A (production).
3. Load the dump onto Server B (shadow).
4. Run dbcc commands (checktable, checkalloc, checkcatalog) on the older release databases just loaded to Server B. 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 version 15.0 (Server B).

Note If you perform an in-place upgrade, so that the older versions of the databases no longer exist, you must create a new Adaptive Server installation for your older server.

If you created a new installation and migrated data using dump and load or similar commands, start the old version of Adaptive Server on the old installation. You may need to use bcp out to copy changed data from 15.0 server back to the older server.

Cutover migration task list example

This sample task list covers a cutover migration approach. 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 but includes the following task lists:

- Set up Adaptive Server 15.0 on development system
- Define test/acceptance criteria—regression test suites
- Define fallback procedures on test system
- “Baseline” older environment on test system
- Run regression test suites on older release test system
- Upgrade test system to version 15.0
- Run regression test suites on version 15.0 Test System
- Run user/acceptance tests on version 15.0 test system
- Execute fallback procedures on test system
- Upgrade production server to Adaptive Server version 15.0
- Perform final steps

Set up Adaptive Server 15.0 on development system

1. Configure the physical drives and local array, duplicating the environment of the current development system.
2. Perform a dump of the current development system.
3. Install Adaptive Server 15.0.
4. Configure Adaptive Server 15.0 on the development system, duplicating the current development environment.

5. Update the interfaces file.
6. Create the databases on Adaptive Server 15.0.
7. Load the dump of the current development system.
8. Run dbcc commands (checktable, checkalloc, checkcatalog) on Adaptive Server 15.0 databases. Check the dbcc log and error log.
9. Synchronize user IDs between the old and new development systems.
10. Run checkpoint on the 15.0 environment.
11. Shift development to Adaptive Server 15.0. 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 code for target user functions.
3. Encapsulate SQL code 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 fallback procedures on test system

1. Create fallback scripts to re-create the older environment.
2. Document fallback procedures.

“Baseline” older environment on test system

Recycle Adaptive Server.

Run regression test suites on older release 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 the 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. Decide whether the test system is ready to be upgraded.
4. Verify and document the regression test.
5. Verify performance and functions on the older release test server.

Upgrade test system to version 15.0

1. Perform a dump of the current system’s 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 version 15.0.
6. Run dbcc commands (checktable, checkalloc, checkcatalog) on the 15.0 databases. Verify the dbcc log and error log.
7. Create a baseline for the 15.0 configuration.

Run regression test suites on version 15.0 Test System

Recycle 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 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. Decide whether the 15.0 system is ready to be upgraded.
4. Verify and document this 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 this regression test.

Other testing

Verify 15.0 performance and functions on the test system.

Run user/acceptance tests on version 15.0 test system

1. Have testers perform user regression testing of version 15.0.
2. Monitor and capture system dynamics (sp_who, sp_lock...).
3. Verify and document the regression test.
4. Decide whether the 15.0 production system is ready to be upgraded.

Execute fallback procedures on test system

1. Shut down the version 15.0 test system.
2. Re-create the older (current production) release test system.
3. Re-create the older environment, using your re-creation scripts and procedures.
4. Load the older production system dumps.
5. Run dbcc commands (checktable, checkalloc, checkcatalog) on the older databases. Verify the dbcc log and error log.
6. Run checkpoint on the older release test system.

Upgrade production server to Adaptive Server version 15.0

1. Dump the databases on the current 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 version 15.0.
6. Run dbcc commands (checktable, checkalloc, checkcatalog) on the version 15.0 databases. Verify the dbcc log and error log.
7. Create a baseling for the version 15.0 configuration on the production system.
8. Perform user testing and verification.

Perform final steps

1. Start production users on the version 15.0 production system.
2. Upgrade the test system to version 15.0.

Staged cutover task overview

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

- The large company has a mature client/server environment with multiple applications residing on a single 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 version 15.0 on the development system, duplicating the earlier release development configuration.
2. Migrate the development objects to the 15.0 development system.
3. Construct regression test suites.
4. Construct bcp scripts to move object deltas.
5. Create a baseling for the older test environment.
6. Configure a duplicate 15.0 server on the test platform.
7. Migrate the test/acceptance objects to the 15.0 test system.
8. Conduct regression test suites on the test system.
9. Verify synchronization of objects between the old and new release test systems.
10. Conduct user acceptance testing on the 15.0 test system.
11. Configure a duplicate 15.0 server on the production platform.
12. Migrate the production objects to the 15.0 production system.
13. Move the production users to the 15.0 production system.

14 Use your bcp scripts to resynchronize objects between the old and new release production systems.

APPENDIX C

Migration Issue Checklists

The following checklists list issues you may want to address in your migration planning. Not all issues shown here will apply to your site, nor are these lists exhaustive. For more information on writing a migration plan, see Chapter 3, “Writing a Migration Plan.”

Logical data architecture

Make sure your logical data architecture includes:

- A graphical representation of the data model from the logical architecture
- A usable map 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 and 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

Verify that the logical application architecture includes:

- A list and brief description of the required RPCs and stored procedures needed to support the new IT architecture
- A list of any potentially sharable functions
- A list of shared services such as initialization, termination, global edits, error handling, logging, and monitoring
- An approach and a graphical representation of how application functionality will be split between servers, clients and middleware
- A list of functional controls that must 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 with which the new IT architecture applications must comply
- A list of services that the new IT architecture applications must interface with, such as image, voice-mail, e-mail, word processing, printing, faxing 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

Make sure the logical technology architecture includes:

- Vendor-independent descriptions (functions and features) and graphical representations 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, and so on), including:
    - Projected quantities
    - Role in the new IT architecture (client, service provider and so on)
    - 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, such as fault tolerance and hot standbys, and so on.

Logical support architecture

Verify that the logical support architecture includes:

- 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

Make sure your migration strategy includes:

- An implementation sequence or a transition plan for candidate application, 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

APPENDIX D Pre-Upgrade Checklist

You can use this checklist in addition to that in the installation guide. It lists steps to prepare for upgrade in order of earliest tasks to tasks done just before upgrading.

Pre-upgrade checklist

1. Install the most recent version of the operating system that has been certified for Adaptive Server.
2. Check that you have enough disk space. See your installation guide for requirements.
3. Check that each database has at least 10% free space. This is needed by the upgrade.
4. Check your applications to assess impact from SQL changes. See Chapter 4, "Making Required Application Changes."
5. Check for new reserved words, including those in scripts and applications. See "New reserved words" in this manual.
6. Replace obsolete environment variables in applications and scripts.
7. Record size and device fragment information for all databases. You can query the sysdevices and sysusages tables for this information.
8. Make a note of the default character set and sort order. See the System Administration Guide at http://manuals.sybase.com:80/onlinebooks/group-as/asg1250e/sag/@Generic__BookView for information about character sets and sort order.
9. Verify that sybsystemprocs is large enough and that all device fragments, if there are multiple fragments, contain both log and data segments. See "sybsystemdb"in Chapter 5, "Making Database Administration Changes."

Just before upgrading:

1 Set master as the default database for “sa”.

2 If the locations of your error log and configuration file are not the default locations, change them to the default location.

3 Back up:

- All databases
- devices
- Sybase directory
- (For Windows) Sybase entries in the Registry

4 bcp out critical system tables:

- syslogins
- sysloginroles
- sysdatabases
- sysusages
- sysdevices

5 On Windows, change the service type for Adaptive Server to “manual.”

6 Disable mirroring.

7 Use sp_auditoption “enable auditing”, “off” to disable auditing.

8 Save the server configuration (*.cfg*) file.

Index