Sybase® SQL Server™ Migration Guide: Moving from Release 10.x or 4.x to Release 11.x

Sybase SQL Server Release 11.x
Document ID: 36066-01-1100-01
Last Revised: October 31, 1996

Principal author: Sybase Technical Support Publications

Document ID: 36066-01-1100

This publication pertains to Sybase SQL Server Release 11.x of the Sybase database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Document Orders

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor.

Upgrades are provided only at regularly scheduled software release dates.

Sybase Trademarks

Sybase, the Sybase logo, APT-FORMS, Data Workbench, DBA Companion, Deft, GainExposure, Gain *Momentum*, Navigation Server, PowerBuilder, Powersoft, Replication Server, S-Designor, SQL Advantage, SQL Debug, SQL SMART, SQL Solutions, SQR, Transact-SQL, and VQL are registered trademarks of Sybase, Inc. ADA Workbench, AnswerBase, Application Manager, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, APT Workbench, Backup Server, Bit-Wise, Client-Library, Configurator, Connection Manager, Database Analyzer, DBA Companion Application Manager, DBA Companion Resource Manager, DB-Library, Deft Analyst, Deft Designer, Deft Educational, Deft Professional, Deft Trial, Developers Workbench, DirectCONNECT, Easy SQR, Embedded SQL, EMS, Enterprise Builder, Enterprise Client/Server, Enterprise CONNECT, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, ExElerator, Gain Interplay, Gateway Manager, InfoMaker, Interactive Quality Accelerator, Intermedia Server, Maintenance Express, MAP, MDI, MDI Access Server, MDI Database Gateway, MethodSet, Movedb, Navigation Server Manager, Net-Gateway, Net-Library, New Media Studio, ObjectCONNECT, OmniCONNECT, OmniSQL Access Module, OmniSQL Gateway, OmniSQL Server, OmniSQL Toolkit, Open Client, Open ClientCONNECT, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerCONNECT, Open Solutions, PC APT-Execute, PC DB-Net, PC Net Library, Powersoft Portfolio, Powersoft Professional, Replication Agent, Replication Driver, Replication Server

Restricted Rights

Table of Contents

About This Book

1. Planning Migration

2. Migrating from Release 10.x to 11.x

3. Migrating from Release 4.x to 11.x

A. Worksheets—Your Current Environment

B. Sample Migration Task Lists

C. Recovery from Upgrade Failure

About This Book

This book, *SQL Server Migration Guide: Moving from Release 10.x or 4.x to Release 11.x*, helps you prepare for and perform migration, as well as execute tests afterward.

Audience

Database and system administrators, as well as application programmers, will find this guide useful.

How to Use This Book

Read chapter 1 for tips on planning, and then read the migration chapter that applies. For example, if your site is migrating from release 4.x to 11.x, go to chapter 3.

The contents are as follows:

Chapter or Appendix	Description
Chapter 1, "Planning Migration"	Outlines three strategies for migration and addresses their advantages and disadvantages. Also gives testing guidelines.
Chapter 2, "Migrating from Release 10.x to 11.x"	Steps you through migration from SQL Server release 10.x to 11.x, detailing 11.x feature changes to consider.
Chapter 3, "Migrating from Release 4.x to 11.x"	Steps you through migration from SQL Server release 4.x to 11.x, detailing 10.x and 11.x feature changes to consider.
Appendix A, "Worksheets—Your Current Environment"	Provides sample worksheets for surveying your current environment.
Appendix B, "Sample Migration Task Lists"	Lists a sequence of common migration tasks.
Appendix C, "Recovery from Upgrade Failure"	Details the steps in troubleshooting when a SQL Server upgrade fails.
Appendix D, "FAQs: Migrating SQL Server 4.x or 10.x on SunOS to 11.x on Solaris"	Answers frequently asked questions about SunOS to Solaris support and migration.

Chapter or Appendix	Description
Appendix E, "FAQs About SQL Server 11.x on Windows NT"	Considers frequently asked questions about SQL Server installation, configuration, and administration on the Windows NT platform.
Appendix F, "Create Database Devices on AIX with Additional Space"	Provides information about accommodating the Logical Volume Control Block on the RS/6000 AIX platform.
Appendix G, "SQL Server 11.x Interoperability and Platform Compatibility"	Lists SQL Server interoperability with other Sybase products, including version numbers and supported platforms.
Appendix H, "ANSI SQL '89 and FIPS 127-1 Standards Compliance"	Lists the changes, as of release 10.x, for full standards compliance.

Related Documents

Here are SQL Server manuals that you are likely to find useful along with this migration guide:

- *What's New in Sybase SQL Server Release 11.0?* summarizes new features.
- SQL Server platform documentation describes SQL Server installation, configuration, and operating system administration tasks.
- *System Administration Guide* covers SQL Server and database administration in depth.
- *Reference Manual* details SQL Server commands and system procedures.
- *Performance and Tuning Guide* explains tuning SQL Server to maximize performance.

Electronic Information Sources

For the most up-to-date information on Sybase products, including information on availability, certifications, bugs, and troubleshooting, refer to Sybase's electronic services:

- AnswerBase™, Sybase's CD-ROM knowledge base
- Sybase OpenLine and PrivateLine, the Sybase forums on CompuServe

- The SupportPlus[SM] Online Services and SyBooks-on-the-Web World Wide Web pages

SupportPlus Online Services and SyBooks-on-the-Web

To get to SupportPlus Online Services and Sybooks-on-the-Web:

1. Connect to the Sybase home page: www.sybase.com.
2. Follow links to Services and Support.
3. Follow links to Sybase Enterprise Technical Support.
4. Follow links to SupportPlus Online Services or SyBooks-on-the-Web.

SyBooks-on-the-Web is accessible to the public.

SupportPlus Online Services are intended for customer use only. Therefore, you must register to access SupportPlus Online Services.

Information Relevant to Migration

You can find current SQL Server 11.x information online.

On the Internet the Sybase home page highlights migration information. The URL is:

`www.sybase.com/products/system11/migration/index.html`

At this web site, you can find such documents as:

- *A Technical Look at Data Partitions with Sybase SQL Server 11*
- *Understanding the Benefits of Multiple Network Engines*
- *Configuring and Tuning Data Cache with SQL Server 11*

From the Sybase home page, you can follow links to the Support*Plus* OnLine Services Technical Information Library for such information as:

- Product documentation under SyBooks-on-the-Web.
- Technical notes. For example, see:
  - *Frequently Asked Questions about bcp*
  - *Tuning Sybase SQL Server for Bulk Loads: Versions 4.9, 10, 11.*
  - *Segment Remapping with* **load database** *When Moving a Database*
- *Sybase Technical News,* issued quarterly.

- White papers. For example, see *Analyzing, Handling and Resolving Optimizer Problems/Symptoms*.

AnswerBase, shipped via CD-ROM, provides similar information to that found in the worldwide web Technical Information Library.

➤ ***Note***

For information about platform-specific commands and issues, see your operating system documentation.

Conventions

The font and syntax conventions in this book are as follows:

Element	Example
Command names, command option names, utility names, utility flags, and other keywords are **bold**.	**installmaster**
Database names, datatypes, file names, and path names are in *italics*.	*sybsystemprocs* database
Syntax that you enter online is in **bold**.	**sp_configure devices, 11**
Variables, or words that stand for values that you fill in, are in ***bold italics***.	***database_name***

If You Need Help

Help with your Sybase software is available in the documentation and from Sybase Technical Support.

Each Sybase installation has a designated person who may contact Technical Support. If you cannot resolve your problem using the documentation, ask the designated person at your site to contact SybaseTechnical Support.

1 Planning Migration

This planning needed to migrate to SQL Server 11.x includes understanding the migration path, assessing your environment, choosing your approach, and building the infrastructure.

The Migration Path

The migration path requires a series of steps which fall into three categories:

- Preparation
- Migration
- Tuning

Preparation

Preliminary steps toward migration include:

- Survey your current environment. Appendix A, "Worksheets—Your Current Environment," provides sample worksheets of the kind of information that is useful in planning for migration.
- Review the SQL Server feature changes for their impact on your applications and system procedures. You can find the details in chapter 2 or 3, whichever is appropriate to your site, under:
  - "Application Changes"
  - "Database Administration Changes"
  - "System Administration Changes"
- Determine the approach you want to take, beginning with the information in this chapter.

Necessary Reading

Before planning and carrying out the migration, become familiar with SQL Server 11.x. Good starting points include:

- The Sybooks guide, *What's New in Sybase SQL Server Release 11.0?*

- Information available online at *www.sybase.com*, such as migration tips from the migration web page and documentation under Sybooks-on-the-Web

Read these documents for important information about upgrading:

- *Release Bulletin* (for your platform)
- The SQL Server installation and configuration documentation for your platform

  If you are upgrading from a release 4.x SQL Server, the installation and upgrade utility will be unfamiliar to you. SQL Server installation instructions explain how to use **sybinit** (UNIX) or **SQL Config** (Windows NT).
- *SQL Server Performance and Tuning Guide*

Migration

The migration process itself has several steps, including upgrading data, testing, and migrating your applications. For details, read the chapter applicable to your site—Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x."

Tuning

This book includes quick tuning tasks, not in-depth performance tuning guidelines.

Release 11.x revolutionalizes SQL Server performance tuning. Enhancing SQL Server 11.x performance typically involves:

- Assessing your performance requirements, database design, and applications
- Identifying SQL Server 11.x features that help improve performance
- Taking advantage of SQL Server 11.x tuning options

For comprehensive information, see the *SQL Server Performance and Tuning Guide*, which covers handling such issues as database design, query optimization, memory use, and *tempdb* performance, as well as monitoring tools.

Survey Your Current Environment

Before creating a migration plan, make sure that you have enough information about the current environment. This migration guide includes several worksheets, which you can use to gather the necessary information. Located under Appendix A, "Worksheets—Your Current Environment," they include:

- *Infrastructure Worksheet*—capture the current production and development environment for analysis and reference
- *SQL Server Operational Worksheet*—record operational business requirements, as well as current SQL Server backup, recovery, and maintenance procedures
- *Data Architecture Worksheet*—survey client application components, process, and data access to a SQL Server database and remote data repository

Determine Your Migration Approach

The migration strategy you use will depend on such factors as the cost of the effort, the size of the database, and available resources.

Select the best approach for your environment as follows:

- Parallel with replication

  Copy pre-release 11 databases to release 11.x databases, and use Replication Server to maintain both sets of databases. The release 11.x system becomes the primary server, and you maintain the pre-release 11.x system as a backup server. This approach is typical for environments that run operations around the clock, 7 days a week / 24 hours a day (7x24).

➤ ***Note***

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

- Cutover

  Change all databases from pre-release 11.x to release 11.x at the same time. A cutover without replication is common in small organizations for development or production servers.

- Phased cutover

  Change only one application and database to release 11.x at a time.

Which Approach is Best?

The following table highlights the advantages and disadvantages of each migration approach. For issues specific to each one, see"Design Your Approach" on page 1-5.

Table 1-1: Migration approaches

Approach	Advantages	Disadvantages	When Used
Parallel with replication	Easy fallback to release 10.0.2.5 or later. You do not need to rebuild release 10.0.2.5 or later databases. Minimal system down time.	Can be complex in OLTP environments. Replication Server must be set up, requiring extra hardware and software. You can replicate from release 4.x, but you cannot replicate to release 4.x.	This approach is best for large 7x24 production databases, maintaining high availability, when: • Rebuilding a release 10.0.2.5 or later database can take too long. • The system may have a large number of transactions and complex Transact-SQL queries with subqueries.
Cutover (no replication)	Can be executed with minimal resource demands.	Highest risk. Requires down time for critical migration tasks. Recovery can be time consuming in a production environment.	This approach is likely for resource-constrained environments.

Table 1-1: Migration approaches (continued)

Approach	Advantages	Disadvantages	When Used
Phased cutover	Low risk with low development overhead. Especially conducive to testing.	May require additional resources—either more memory or a second system, if both pre-release 11.x and release 11.x servers do not perform acceptably on the current system. Requires tighter coordination with application groups and database owners.	If neither of the other two approaches seems appropriate, you can use a phased cutover.

➤ ***Note***

For any migration, first, change only **test** and **development** databases from pre-release 11.x to release 11.x. After significant testing with satisfactory results, move the **production** databases.

Design Your Approach

Specific issues to address in preparing for migration include:

- Fallback—Plan what to do in case the migration fails.
- Application test suite—Determine what validation and performance testing to perform for acceptance.
- Bridging—Specify ways to minimize the impact to users during the migration.
- Environment—Identify what additional resources are required and what changes to the environment are needed.
- Scheduling—Forecast how much time the migration will take based on the level of complexity.

➤ ***Note***

For a task list example of each approach, see Appendix B, “Sample Migration Task Lists.”

Parallel

The issues for the parallel (with replication) approach are:

Table 1-2: Parallel migration

Issue	Recommendations and Tips
Fallback	Plan for all users to reconnect to SQL Server 10.0.2.5 or later after you take SQL Server 11.x offline. Make TCP/IP address and port changes where appropriate. Test the fallback process as part of the application test suite. This suite should do both of the following: • Insert data into SQL Server 11.x. The data must be replicated and available in SQL Server 10.0.2.5 or later. • Execute the fallback script. Consider making daily **bcp** dumps of the databases. To fall back, you can then load the dumps into release 10.x. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**. For information about scheduling backups of user databases, see the *System Administration Guide*. Here are some additional tips: • Make sure to upgrade Replication Server first. • Make sure the applications are using the correct server. For details on the interfaces file and the $DSQUERY environment variable, see the configuration guide for your platform. • Remember to include any client fallback in the calculation. • For 7x24 sites, load time delays can impact synchronization. Consider replicating to SQL Server 11.x and then switching servers.
Application test suite	Since the parallel with replication approach is best for high availability applications, it is imperative that the test suite addresses both update correctness and performance acceptability. Execute this suite *before* switching users to the new system. Also see “Write the Application Test Suite” on page 1-14. **Note:** After successful validation, consider having users enter production queries with the production toolset. A good time to do so is after hours or during production lulls.

Table 1-2: Parallel migration (continued)

Issue	Recommendations and Tips
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues. To ensure correct updates and acceptable performance, test the replicated environment. See the *Replication Server Administration Guide*.
Environment	The environment used for SQL Server 11.x may need to be more powerful to handle the query and replication loads. See the *Replication Server Configuration Guide*. Be sure to account for any increased release 11.x memory requirements that apply to your configuration. For more information, see: • "Setting Up Memory and Cache for an Upgrade" in chapter 2 or 3 (whichever is appropriate to your environment) • Details on configuring memory and data caches in the *System Administration Guide* **Note:** For a production system, execute the performance suite during off hours.
Scheduling	Developing and running the replication facility, validation and preformance suite, and fallback script requires significant effort. If your environment already uses replication, this effort will be notably easier. For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Cutover

The issues for the cutover without replication approach are:

Table 1-3: Cutover migration

Issue	Recommendations and Tips
Fallback	Base fallback on the amount of time needed to restore release 10.x or release 4.x. For example, if users need the system Monday at 8 a.m. and the restoration takes 8 hours, the validation test must pass by Sunday at midnight. **Note:** Remember to include any client fallback in the calculation. You can use **dump database** or **bcp..out** before an upgrade to prepare for fallback. Plan a way to capture transactions after cutover to be used in case of fallback. If you go into production and then need to back off, you will have to restore all the transactions that occurred after the last dump/load.
Application test suite	For a development system, simple validation may be adequate. However, for a **production** system, it is imperative that the test suite addresses both update correctness and performance acceptability. **Note:** You might want to validate over a 3-day weekend if possible.
Fallback after cutover	Consider making daily **bcp** dumps of the databases. You can then load the dumps into release 10.x to fall back. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**. For information about scheduling backups of user databases, see the *System Administration Guide*.
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues.
Environment	Be sure to account for any increased release 11.x memory requirements that apply to your configuration. For more information, see: • "Setting Up Memory and Cache for an Upgrade" in chapter 2 or 3 (whichever is appropriate to your environment) • Details on configuring memory and data caches in the *System Administration Guide* **Note:** For a production system, execute the performance suite during off hours.
Scheduling	For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Phased Cutover

The issues for a phased cutover are:

Table 1-4: Phased cutover migration

Issue	Recommendations and Tips
Fallback	Consider making daily **bcp** dumps of the databases. To fall back, you can then load the dumps into release 10.x or 4.x. Keep in mind: • You may need to modify the databases to support incremental **bcp** dumps. • Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. • Do not use release 11.x schema enhancements, such as **max_rows_per_page**, declarative RI (referential integrity) and IDENTITY columns, until the conversion succeeds. For information about scheduling backups of user databases, see the *System Administration Guide*.
Application test suite	Ensure that the application test suite addresses both update correctness and performance acceptability. Also ensure that you do the following: • Maintain the directories / libraries for both releases. • Make sure the applications are using the correct server. • After successful validation, consider having users enter production queries with the production toolset. A good time to do so is after hours or during production lulls.
Bridging	There should not be any user impact during migration. The more stringent the validation test is, the less likely you will have bridging issues. Pre-release 11.x cannot read release 11.x backup files. You need to create **bcp** or other scripts to move tables back to pre-release 11.x. **Note:** Do not use release 11.x schema enhancements, such as **max_rows_per_page**, declarative RI and IDENTITY columns, until the conversion succeeds.
Environment	Be sure to account for any increased release 11.x memory requirements that apply to your configuration. For more information, see: • "Setting Up Memory and Cache for an Upgrade" in chapter 2 or 3 (whichever is appropriate to your environment) • Details on configuring memory and data caches in the *System Administration Guide* Here are some additional tips: • Execute performance measurements on a system with similar capabilities. • For a production system, execute the performance suite during off hours.

Table 1-4: Phased cutover migration (continued)

Issue	Recommendations and Tips
Scheduling	For a **development** system, you may want to add a short period to the development schedule for release 11.x issues. For a **production** system, be prepared to postpone or back off if needed.

Build a Migration Infrastructure

Before you begin the migration to release 11.x, be sure that you have the tools appropriate for your migration approach, including:

- Hardware resources
- Proper operating system version and fix level
- Proper revision levels for Sybase products
- Testing plan and scripts to cover functionality, stress, and operational testing

The absence of any of these tools could result in a delayed migration or a troubled, back-and-forth migration.

Hardware Resources

Evaluate hardware resource needs according to your migration approach, such as parallel with replication. You may need a separate system, Replication Server for the high availability environment, a disk farm for backout strategies, and more memory. Use components for the test system that are identical to the production system, such as operating system, drivers, and disk type.

If the test system is smaller than the production system, consider the following:

- Less disk space requires that you scale down the database as follows:
  - Retain the same data distributions for consistent optimizer decisions.
  - Scale down memory correspondingly to ensure consistent I/O rates.
  - Make the data layout across devices as close to that of the production system as possible.

- For fewer CPUs, adjust the transaction load or concurrent users to account for less potential work.

For more information about memory and cache requirements, see the applicable chapter:

- Chapter 2, “Migrating from Release 10.x to 11.x”
- Chapter 3, “Migrating from Release 4.x to 11.x”

Proper Operating System Version and Fix Level

Ensure that the operating system is at the proper version and level to run SQL Server 11.x. For migration across machines, verify that the release 10.x or release 4.x server will run under the same operating system level.

For release 11.x operating system requirements, see the *Release Bulletin* for your platform.

This guide includes some platform specific information as follows:

- If your current SQL Server resides in a SunOS environment, see Appendix D, “FAQs: Migrating SQL Server 4.x or 10.x on SunOS to 11.x on Solaris.”
- For Windows NT tips, see Appendix E, “FAQs About SQL Server 11.x on Windows NT.”
- To allow enough space for the RS/6000 AIX Logical Volume Control Block, see Appendix F, “Create Database Devices on AIX with Additional Space.”

Review SQL Server Interoperability with Other Sybase Products

To ensure that the versions of other Sybase products in use at your site are compatible with SQL Server 11.x, see Appendix G, “SQL Server 11.x Interoperability and Platform Compatibility.”

Testing Plan and Scripts

Plan to test SQL Server 11.x in a test environment before migration to your production environment. Appendix B, “Sample Migration Task Lists” provides sets of steps for each migration approach. The high-level tasks for any migration include:

1. Create / upgrade your **test** system.
2. Make necessary application changes.
3. Iteratively run test suites.
4. Move to a **production** system.
5. Run operational and acceptance tests.

Use your production environment to model a test system; that is, have a representative multiuser workload and do profile tests. In designing and running tests, be sure to do the following:

- "Baseline" all tests with the pre-release 11.x system.
- Include backup, fallback, preventative maintenance, and system housekeeping.
- Examine the results to pinpoint expected behavior.

In defining and building your test environment, ensure the following:

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

Use application test suites and performance scripts to do your analysis. Be sure to prioritize critical processes, based on user-defined priorities, and build your test suites for prioritized application functions. The information you gathered in Appendix A, "Worksheets—Your Current Environment," such as the transaction profile, is useful in designing tests.

Develop "Reset" and "Sanity Test" Procedures

In preparation for testing, you may want to develop the following procedures:

- "Reset" procedures for backing out of changes during simple testing; that is, test runs where the environment is in an unknown state and a restore from backup is required.
- "Sanity test" procedures to check:
  - Query plans. You can run each transaction to capture **statistics io**, **statistics time**, **showplan**, and **dbcc traceon(302, 310)** output. When

comparing the response time and task context switching in pre-release 11.x and release 11.x tests, keep in mind that release 11.x query plans are much larger.

For information about these functions, see both the *SQL Server Reference Manual* and *Performance and Tuning Guide*. For information about analyzing an optimizer problem, see the *Troubleshooting Guide*.

- Indexes. Compare the test index definitions with production versions. For more information about indexes, see the *Transact-SQL User's Guide* and the *Performance and Tuning Guide*.

Identify Response Time Problems

For response time problems:

1. Check query plans, comparing fresh **showplan** output with that of a known version. For more information about:
   - Description of **showplan** output, see the *Performance and Tuning Guide*
   - Interpretation of **showplan** output, see the white paper *Analyzing, Handling and Resolving Optimizer Problems/Symptoms* (available on the Web in the Technical Information Library)
2. Check indexes. For information on fixing corrupted indexes, see the *Troubleshooting Guide*.
3. Check for lock contention using **sp_who**. For more information, see the *Performance and Tuning Guide*.

Establish Baseline on the Current SQL Server

Gather information on your current environment to establish a baseline for release 11.x testing. Useful steps include:

1. Set up the test databases.
2. For a clean startpoint, check the consistency of pre-release 11.x databases that you build from production backups. Run **dbcc** with options **checkdb**, **checkalloc**, and **checkcatalog**.

   For details on checking database consistency, see the *System Administration Guide*.
3. Back up the databases.
4. Run single-user tests to validate optimizer decisions as follows:

- Capture **showplan**, **statistics io**, and **statistics time** output.

  If you find inconsistencies, investigate further by running **dbcc traceon(302,310)**. For details on using trace flags 302 and 310, see *Analyzing, Handling and Resolving Optimizer Problems/Symptoms*. This white paper is available in the Sybase Technical Information Library on the Web.

- Compare the output to that of the production system. The optimizer decisions should be the same, and the counts of logical and physical I/O should have the same ratio as the test and production database sizes.

- Fix any query plan problems that you discover. Otherwise, you cannot closely compare the test and production systems, and any multiuser test will be invalid. For more information, see the *Performance and Tuning Guide* and the white paper *Analyzing, Handling and Resolving Optimizer Problems/Symptoms*.

5. Iteratively run multiuser tests as follows:

   - Execute test scripts and capture the transaction throughput and response time metrics, as described in step 2.

   - Fix bottlenecks and tune problems, such as different cache hit rates between tests.

   - Restore the databases to a known state, reloading them from backup. For details on backing up user databases, see the *System Administration Guide*.

   - Rerun the tests until you are satisfied.

Write the Application Test Suite

The testing scripts for applications should include rigorous subquery testing to validate that the subqueries return the same results in SQL Server 11.x as in the version you have been using. You may have a script available already, if you test application enhancements.

For migration from release 4.x servers, it is important to understand the changes to subqueries in both release 10.x and release 11.x. For more about subquery changes, see Chapter 3, “Migrating from Release 4.x to 11.x.”

Write the Performance Script

A performance script enables you to determine whether immediate performance goals are satisfied. For meaningful goals, consider the following, based on your migration approach:

- For **parallel with replication**:
  - Be sure to measure the overhead of the replication mechanism. For example, if the replication costs 10%, do not begin parallel operations until release 11.x 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**, you gear the migration for equivalent performance between the old and new systems. A goal of breaking even the first week is reasonable.
- A **phased cutover** is subject to the highest performance expectations. Some performance tuning of the production workload after cutover of the production server may be best. You can time the production server cutover to occur as soon as performance gains are acceptable and system / operational testing is successful.

If you are constructing a performance script, consider the following:

- 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.
- If workload is based on client PCs issuing Transact-SQL, you can use performance monitoring tools, available in the market, to capture datastreams with “recorder” or “sniffer” products.
- Volume is critical in performance simulation. A script roughly equivalent to your applications, running at the same volume of queries that an application generates, is usually better than a script functionally matched to an application, but running at only half the volume.

Consider Alternative Testing

Testing techniques to consider along with your existing application test suites, include the following:

Table 1-5: Alternate testing

Technique	Description	Advantages	Disadvantages
Ad hoc test	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 test scripts	Specify input and compare with known outputs.	Easy to implement. Common basis for regression test suites.	Only tests back- end server. **Note:** Back-end focus may help locate the cause of a problem. Impossible to obtain production multiuser load, which misses concurrency and capacity issues altogether. No ad hoc query testing. Depends on strong analysis of process or transaction profiles.
Keystroke capture tools	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.
Transaction generator	Simulate user execution of transactions.	Strong multiuser load testing. Focus on back-end server issues.	May increase development time for creating multiuser test simulations, and add time for debugging test harness. Depends on strong analysis of process or transaction profiles.

Table 1-5: Alternate testing (continued)

Technique	Description	Advantages	Disadvantages
Production load capture	Capture real transactions in a production environment, including their performance and semantic characteristics. Resubmit in a test environment for analysis.	Tests real production loads, including ad hoc queries. This is 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.

Stages of Testing

Testing should be rigorous, employing the techniques just described in several stages. Each stage and its focus is as follows:

Table 1-6: Testing stages

Stage	Purpose	Best Technique
Functional testing	For each application or process, test for: • Good execution of critical transactions • Expected results from transactions • No application or process breakage • Support of functionality in this release • New features in this release	Single-user: • Ad hoc test • Manual test scripts and cases • Existing application test suites
Stress testing	Use production level loads to test for: • Bugs related to multiuser loads • Good execution of critical transactions • Stability of the new release	Multiuser: • Keystroke capture • Transaction generator • Production load capture

Table 1-6: Testing stages (continued)

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

Migration Checklist

The following checklist, based on the migration chapters in this book, shows the release 11.x impact on pre-release 11.x environments.

Table 1-7: Migration checklist

	Task	For Migration from Release 10.x	For Migration from Release 4.x	For more information
❒	Update database load procedures. Add **online database** command to load script.	X	X	See the topic about databases online/offline in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Update scripts that use **sp_configure**.	X	X	See the topic on the new approach to storing configuration values in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Rewrite certain subqueries. **Note:** Remember to test all subqueries before production database transfer.	X	X	See the topic on queries in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Change database and object names that conflict with new keywords.	X	X	See the topic on new keywords in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Increase size of system stored procedures database.	X	X	See the topics on *sybsystemprocs* in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.

Table 1-7: Migration checklist (continued)

	Task	For Migration from Release 10.x	For Migration from Release 4.x	For more information
❒	Update dump procedures as needed: • Remove pointers to */dev/null* or *NL:*. • Accommodate multiple dumps to single tape. • Change user-defined transactions that dump databases or transactions.		X	See "Changes to Dumps" in Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Decide and test "last chance threshold" behavior.		X	See "Decide Last Chance Threshold Behavior" in Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Change queries as needed.		X	See the following topics in Chapter 3, "Migrating from Release 4.x to 11.x": • "Comment Protocol" • "Transact-SQL Syntax" • "No Null Column Headings" • "Correlation Name Consistency"
❒	Change backup procedures as needed.		X	See the following topics in Chapter 3, "Migrating from Release 4.x to 11.x": • "Backup Server" • "No dump transaction to Device After Non-Logged Text Writes" • "New Database for Stored Procedures"
❒	Change your system startup file to use the renamed run file and the new configuration file.		X	See the following topics in Chapter 3, "Migrating from Release 4.x to 11.x": • "Run File Renamed" • "New Approach for Storing Configuration Values"

Table 1-7: Migration checklist (continued)

	Task	For Migration from Release 10.x	For Migration from Release 4.x	For more information
❒	Make changes to support the new system stored procedures database.		X	See "sybsystemprocs Database" Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Review changes to output from system stored procedures.	X	X	See the topic on system storec procedures output in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Check error message text within applications.	X	X	See the topic on improved error messages in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Ensure successful *master* database backup.		X	See "Loading from Multiple Tape Devices" in Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Prevent dump failure on first dump after upgrade.	X	X	See the topic on Backup Server in Chapter 2, "Migrating from Release 10.x to 11.x" or Chapter 3, "Migrating from Release 4.x to 11.x" as applicable.
❒	Review the new datatype, datatype hierarchy, and conversions.		X	See "Datatypes" in Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Review **set arithabort/arithignore** behavior.		X	See "set arithabort / arithignore Behavior" in Chapter 3, "Migrating from Release 4.x to 11.x."
❒	Make sure that scripts using **sp_password** create passwords of at least 6 bytes. Also add roles appropriate to your site.		X	See "New Login and Password Protocols" in Chapter 3, "Migrating from Release 4.x to 11.x." For an understanding of release 11.x security features, see the *Security Administration Guide* and the *Security Features User's Guide*.

2 Migrating from Release 10.x to 11.x

Read this chapter if you are migrating to release 11.x from SQL Server release 10.x.

For release 4.x, see Chapter 3, "Migrating from Release 4.x to 11.x."

This chapter covers the application and administrative changes that are critical to migration from release 10.x, as well as other considerations and recommended testing. The topics include:

Upgraded Server versus Rebuilt Server

For migration to SQL Server 11.x, you perform either an upgrade or rebuild process as follows:

Table 2-1: SQL Server upgrade versus rebuild

Process	Purpose	How	Advantages	Disadvantages
Upgrade	"Promote" databases from release 10.x to 11.x.	Use the install/configure utility for your platform: • **sybinit** (UNIX) • **Server Config** (Windows NT) For more information about using the install/configure utility, see installation and configuration instructions for your platform.	Simple to implement. Upwardly compatible load solution. Low risk.	Limited flexibility for modifying environment during the upgrade.

Table 2-1: SQL Server upgrade versus rebuild (continued)

Process	Purpose	How	Advantages	Disadvantages
Rebuild	Construct a release 11.x environment, then apply your release 10.x environment.	Use one of the following: • **dump/load database** functions. First rebuild the *master* database, either manually or from scripts, and then load the other databases. • **bcp** and **defncopy** utilities. **defncopy** supports defaults, rules, triggers, views, and stored procedures, but not tables, indexes, constraints or declarative referential integrity. • Locally developed scripts or programs.	An opportunity to improve system design. More flexibility for environment modifications.	Resource intensive, including time, hardware, staff. Potentially complex and thus can introduce risk.

Prepare for Migration

This section assumes you have done the necessary planning, outlined in Chapter 1, “Planning Migration,” and you have installed the necessary hardware and software, such as Replication Server.

Summary of Steps

The following is a summary of the SQL Server installation steps. Complete these steps, detailed in the installation instructions for your platform, using the information in this chapter where indicated.

Table 2-2: Summary of installation steps

Phase	Tasks
Planning	1. Read the *Release Bulletin*. 2. Create a “sybase” account to perform installation and configuration tasks. 3. Locate your Customer Authorization String (CAS) on the CD-ROM or tape package.

Table 2-2: Summary of installation steps (continued)

Phase	Tasks
Preparing for Installation	Install a new SQL Server 11.x if you are following a rebuild plan. You may also decide to install a new 11.x server so that you can run tests against it. The following is the sequence of tasks to prepare for an install. For complete details, refer to the installation instructions for your platform. 1. Create the Sybase installation directory. 2. Set the SYBASE environment variable to the path of the Sybase installation directory. 3. Verify permissions on your Sybase installation directory. 4. Determine where to install the new release. 5. Back up the existing Sybase installation directory. 6. If installing other Sybase products, determine which ones should be running. 7. Consider product version and compatibility issues. 8. Make sure that your operating system supports the products you want to install. Also see “Prepare the Platform” on page 2-6. 9. Make sure that your operating system configuration and available disk space are adequate to complete the installation. Also see “Setting Up Memory and Cache for an Upgrade” on page 2-6. 10. Install operating system kernel patches. Also see “Apply Prerequisite Platform Fixes” on page 2-5. 11. Verify your network software configuration. 12. If you are installing remotely from tape, be sure that access permissions are configured correctly. 13. For UNIX platforms, be sure the $DISPLAY environment variable is set to the computer on which you are running **sybsetup**. 14. Make sure that your operating system configuration is adequate to run SQL Server. 15. Set shared memory parameters. 16. For UNIX platforms, enable asynchronous disk I/O. 17. Edit the default login file, if needed. 18. Choose database devices.
Installing	1. Complete the Configuration and Upgrade Worksheet in the installation guide for your platform. 2. Follow the step-by-step installation instructions for your platform.

Table 2-2: Summary of installation steps (continued)

Phase	Tasks
Preparing for the Upgrade	These upgrade tasks apply if you are following a plan to upgrade your server in place, or if you upgrade an existing test server. For complete details on these steps, refer to the installation instructions for your platform. 1. Before upgrading a SQL Server that contains replicated databases, follow the instructions for a replicated system in your installation guide. 2. Make sure that you installed the 11.x software in a different directory from your previous SQL Server installation. 3. Identify the name and location of the runserver file. 4. If you have not already done so, record your configuration. The installation guide provides a worksheet. 5. Test and change your current applications and stored procedures, as needed. • For guidelines on testing see Chapter 1, “Planning Migration.” • For information on potential release 11.x impact on your applications, see “Application Changes” on page 2-13. 6. Determine whether your current SQL Server supports upgrade to release 11.x. • You can upgrade Sybase SQL Server 4.9.2 and higher. • You can upgrade Microsoft SQL Server 4.2. Follow the special instructions in the installation guide for Windows NT platforms. 7. Check for reserved word conflicts. Also see “Check for Reserved Words Before Going “Live”” on page 2-9. 8. Make sure that all SQL Server users are logged off. Also see “Log Off Users” on page 2-9. 9. Check database integrity. Also see “Run dbcc Options” on page 2-10. 10. Back up all databases on your current SQL Server. 11. Be sure the *sybsystemprocs* database is the correct size. 12. Turn off databases options. “Turn Off Options in All Databases but tempdb” on page 2-8. 13. Disable disk mirroring. 14. Configure memory resources. 15. Note the current cache size.
Upgrading	Perform the actual upgrade. For details, see the following information: • Step-by-step instructions in the installation guide for your platform • “Run sybinit (UNIX) or Server Config (Windows NT)” on page 2-10 **Note:** Remember to install Backup Server.

Table 2-2: Summary of installation steps (continued)

Phase	Tasks
Followup	The following is the sequence of tasks to complete after an install / upgrade. For complete details on these steps, see the installation instructions for your platform.
	1. Verify your new SQL Server version.
	2. Reset configuration options.
	3. Reset database options.
	4. Update your scripts. For information on potential release 11.x impact on your scripts, see "Database Administration Changes" on page 2-20 and "System Administration Changes" on page 2-21.
	5. Increase the procedure cache. Also see "Queries" on page 2-13.
	6. Remirror devices.
	7. Drop and recreate procedures with subqueries. Also see "Queries" on page 2-13 and "Drop and Recreate Compiled Objects Containing Subqueries" on page 2-16.
	8. Reenable and restart replication.
	9. Install the new Backup Server if you have not already done so.

Prepare the Environment

The topics in this subsection highlight a few tasks in the SQL Server installation instructions. Before performing them, be sure that you have already completed both of the following:

- Instructions on preparing for a new installation or an upgrade
- The worksheet in the installation guide for your platform

Apply Prerequisite Platform Fixes

Apply prerequisite platform / operating system fixes, based on the *Release Bulletin* and support structure for the platform. If applicable, do the following:

- For an upgrade, test the SQL Server 10.x with any new operating system patches. If the test is successful, you will be able to run release 10.x and 11.x on the same operating system.
- Avoid changing the operating system and SQL Server at the same time, unless you are building a new system. Instead, upgrade the operating system first, stabilize it, and then upgrade SQL Server.

Prepare the Platform

Make platform-specific preparations as follows:

Platform	Action Before Using *sybinit* or *Server Config*
RS6000 / AIX	To install or upgrade SQL Server on RS6000 / AIX, do the following: • Allow 1MB for Logical Volume Control Block. For more information, see *Appendix F, "Create Database Devices on AIX with Additional Space."* • For an install only, check the System Management Interface Tool (SMIT) asynchronous input / output parameters. For more information, see SQL Server installation instructions and your operating system documentation.
SunOS	Before upgrading a SQL Server currently on SunOS, migrate to Solaris according to the release 11.x installation instructions. Also see Appendix D, "FAQs: Migrating SQL Server 4.x or 10.x on SunOS to 11.x on Solaris."

Setting Up Memory and Cache for an Upgrade

Changes in SQL Server 11.x have increased its memory requirements, such as:

- New user log cache
- Larger **dataserver** binary
- Previously existing structures which are now larger (such as locks)

To avoid upgrade problems based on limited memory, perform both of the following steps before the upgrade:

1. **Record the current cache sizes.**

   Look at the error log to get a profile of the memory usage on your production SQL Server, including the buffer cache, procedure cache, and procedure header sizes.

   The following is an example of memory-related messages from the error log for SQL Server with **total memory** parameter of 7500:

```
server: Number of proc buffers allocated: 556
server: Number of blocks left for proc headers: 629
server: Memory allocated for the default data cache: 4144 Kb
```

For an upgrade to release 11.x, you set memory requirements for the upgrade itself, and then use these recorded values as a baseline for determining requirements after the upgrade.

For more information about configuring memory and memory-related messages in the error log, see the *System Administration Guide*.

2. **Configure memory resources.**

   Because of extra overhead requirements, the upgrade process can run out of memory if you do not increase it. You can increase memory in one of two ways:

   - Raise the amount of allocated memory by changing the value of the **total memory** configuration parameter.
   - Lower certain parameters, such as **user connections**, to as close to the default values as possible.

     The defaults for some configuration parameters that consume memory are:

Configuration Parameter	Default Value
user connections	25
locks	5000
open objects	500

For post-upgrade considerations, see “Reconfigure Memory for Normal Usage” on page 2-25.

Tasks Specific to Upgrading SQL Server

If you are upgrading SQL Server 10.x to SQL Server 11.x, perform the tasks in this section.

If you are rebuilding SQL Server, see “Tasks Specific to Rebuilding SQL Server” on page 2-11.

Increase *sybsystemprocs*

Since SQL Server 11.x includes many new system stored procedures, the *sybsystemprocs* database must be larger. Before beginning the upgrade, increase the size of *sybsystemprocs*. Check the *Installation*

and Configuration Guide for the correct size. If *sybsystemprocs* is not the correct size, **sybinit** or **Server Config** will fail.

Disable Replication

Disable replication and clear the log before upgrading to release 11.x. If replication is enabled and your log has not been cleared, the upgrade will fail. For more information:

- The installation guide for your platform details how to handle the Replication Server with a SQL Server upgrade, including the steps necessary to disable replication.
- The *Replication Server Commands Reference* details commands.
- The *System Administration Guide* details how to clear logs.

Turn Off Options in All Databases but *tempdb*

Turn off all options in all databases, except *tempdb*, using **sp_dboption**. The upgrade requires that you set **select into/bulk copy** to true for *tempdb*.

➤ ***Note***

Now is a good time to reconsider your database option settings, if desired.

To turn off a database option, use **isql** to enter the following commands:

```
sp_dboption database_name, "option", false
go
use database_name
go
checkpoint
go
```

To set **select into/bulk copy** for *tempdb*, use the following command:

```
sp_dboption tempdb, "select into/bulkcopy", true
go
use tempdb
go
checkpoint
go
```

See the *SQL Server Reference Manual* for information on **sp_dboption**.

Check for Reserved Words Before Going "Live"

We strongly recommend that you check for the use of reserved words before you upgrade a **production** system. Check for reserved words when you test applications and resolve the conflicts at that time. For details, see "Avoid Keyword Conflicts" on page 2-18.

The **sp_checkreswords** stored procedure checks for reserved word conflicts. Change database names that conflict with reserved words immediately. Otherwise, the upgrade will fail.

To find reserved word conflicts, run the **sp_checkreswords** stored procedure as follows:

Platform Type	To Run *sp_checkreswords*	For More Information
Windows NT	**Server Config** installs **sp_checkreswords** and performs reserved word checking during upgrade.	Installation instructions for Windows NT *Reference Manual* under **sp_checkreswords**
UNIX	**sybinit** installs **sp_checkreswords** automatically. You can use **sybinit** to run **sp_checkreswords** at any time, including before an upgrade, as follows: 1. Begin a **sybinit** session, as described in the installation guide. 2. Select "Check for reserved word conflicts" from the SQL Server Upgrade menu.	Installation instructions for your UNIX platform *Reference Manual* under **sp_checkreswords**

Log Off Users

Immediately before proceeding with the migration, be sure that:

- SQL Server is running.
- All users are logged off and applications that connect to SQL Server are not running.

  Use **sp_who** to determine what users and processes are online.

Run *dbcc* Options

To ensure healthy databases, run the following **dbcc** options on all pre-release 11.x databases:

- **checkdb**
- **checkalloc**
- **checkcatalog**

See the *System Administration Guide* for instructions on running **dbcc**.

Run *sybinit* (UNIX) or *Server Config* (Windows NT)

For instructions on using **sybinit** or **Server Config**, see the SQL Server installation instructions for your platform.

Additional points are:

- Make sure that *sybsystemprocs* size accommodates the system procedure **sp_sysmon** and the system procedures that support **sp_thresholdaction**.
- For a log of the upgrade process:

Platform	Install/Upgrade Log Location	How Created
Windows NT	<*install_path*> \ *init* \ *logs* \ log <*mmdd*>.xxx	The installation process automatically creates the log.
UNIX	*$SYBASE/init/logs/log* <*mmdd*>.xxx	Type **Ctrl w** at the end of a **sybinit** session to retrieve a file of all the selections that you made during the session.

➤ ***Note***

If your upgrade attempt is unsuccessful, see Appendix C, "Recovery from Upgrade Failure."

Tasks Specific to Rebuilding SQL Server

If you are building a SQL Server 11.x environment, perform the following tasks. This section assumes that you have already completed the tasks under "Prepare for Migration" on page 2-2.

Before You Begin

Run **sybinit** (UNIX) or **Server Config** (Windows NT) to install SQL Server 11.x, as described in the installation instructions for your platform.

Check for Reserved Words

You can check for reserved words in either of the following ways:

- Run **sybinit** or **Server Config** against your pre-release 11.x SQL Server, as described in "Check for Reserved Words Before Going "Live"" on page 2-9.
- Use **isql** to run the **installupgrade** script found in the release 11.x *$SYBASE* directory under *scripts*.

Create Databases

Move the data to SQL Server 11.x databases using the **dump/load database** commands or the **bcp** utility.

Building *master*

You cannot use the **load database** command for the *master* database. Instead, build *master*. For details see the *System Administration Guide*. In summary, the steps are:

1. Create devices.
2. Create databases, making sure that they are layed out exactly the same as the old databases.

   For details on loading an old database onto a new one, see the TechNote, *Segment Remapping with* **load database** *When Moving a Database.*
3. Use a script to add users to the *syslogins* table in a consistent order.

For information about reestablishing SQL Server logins after you add users, see the *Troubleshooting Guide*.

4. Add the roles appropriate to your site.

   For more information, see the *Security Administration Guide*.

5. Configure your server.

Using *dump/load database*

To dump and load release 10.x databases:

1. Use the **dump database** command to back up all your databases, except *master*.
2. Bring over one database at a time from SQL Server 10.x to your new SQL Server 11.x, using **load database**.

   You do not need to move the dumps from the release 10.x machine. The remote Backup Server can be release 10.x.

   For details on using the **dump/load database** commands, see:

   - *Reference Manual*
   - *System Administration Guide*, under backing and restoring databases.

Initialize Object Allocation Map Pages

Run **dbcc checkalloc** after you move the data to initialize the Object Allocation Map (OAM) extent pages. This activates direct OAM addressing, one of the SQL Server 11.x speed enhancements.

For details on Object Allocation Map pages, see the *Performance and Tuning Guide*.

Resolve Cross-Database Calls

Move all databases to ensure cross-database referential integrity. If you move some but not all databases, you must drop and re-create stored procedures with cross-database calls.

Using *bcp*

1. Use existing scripts, SQL Server Manager, or third party tools to create:

   - Database objects

- User stored procedures
- Triggers
- Views
- Users
- Permissions

You can also use **defncopy** to create user stored procedures, triggers, and views.

2. Move the data using the **bcp** utility. **bcp** is much faster than it was in release 10.x. Partitioning your target tables, combined with the use of multiple **bcp** streams, makes moving data even faster.

➤ ***Note***

bcp automatically initializes OAM pages. You do not need to run **dbcc checkalloc**, as you do with **dump/load database**.

For more information about:

- Using **bcp**, see the utilities manual for your environment
- Partitioning tables, see *Performance and Tuning*

Application Changes

Before upgrading your **production** system, review this section. Consider testing to highlight potential problems.

This section describes changes to queries and keywords that may impact your applications. It also notes some changes that may be visible through applications after migration.

Queries

This section describes changes to query processing and subquery performance, as well as guidelines for testing subqueries.

New Most-Recently-Used Plan

A new strategy known as **MRU** (most recently used or fetch-and-discard) is available to the optimizer. The MRU strategy is designed to reuse the same buffers in cache and to avoid overwriting all existing pages.

For example, on SQL Server releases prior to 11.x, users running small transactions often find the necessary pages in cache (avoiding physical I/O). When another user runs a **select *** query from a very large table, the cache is overwritten with pages from this table. This forces the other users' queries to use more physical I/O because their pages are no longer in cache.

The release 11.x optimizer can choose either LRU (least recently used) or MRU. To see your optimizer's plan, execute **set showplan on** and then run your query. If you see LRU, your SQL Server is running with old behavior; if you see MRU, it is running with the new 11.x behavior.

You can override the MRU plan if you are unhappy with its performance by using the **lru** option in queries, described in the section on **select** in the *SQL Server Reference Manual*. To get 10.x behavior for your queries, construct it as follows:

```
select * from table (index table prefetch 2 lru)
```

For details on advanced optimizer techniques, see the *Performance and Tuning Guide*.

Subquery Performance

Subquery handling has been improved. For most subqueries, SQL Server 11.x performs better than 10.x. Test all your subqueries under SQL Server 11.x before transferring production databases to the new system.

➤ ***Note***

If you notice performance degradation after migration, check your data cache and increase its size if necessary. SQL Server 11.x needs more memory than 10.x, and will take memory from the default data cache if it does not have enough, causing the cache to shrink. For more information, see "Memory and Cache" on page 2-24.

Expression Subqueries

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

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

- The subquery contains an aggregate.

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

```
select * from huge_table where x=
    (select sum(a) from tiny_table
    where b = huge_table.y)
```

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

```
select huge_table.y, s=sum(a)
    into #t
    from huge_table, tiny_table
    where b=huge_table.y
    group by huge_table.y

select huge_table.*
    from huge_table, #t
    where x=#t.s
    and huge_table.y=#t.y
```

No set dup in subquery

The **set dup in subquery** command, introduced in SQL Server 10.x, is no longer supported. Applications that use it receive a warning message and subqueries no longer return duplicates.

You may have used this option to obtain better performance. Because of release 11.x subquery processing changes, if you want duplicates, rewrite your query as a join. You should see better performance in SQL Server 11.x for these types of queries.

union Limitations

The following limits apply to the **union** operator:

- Only 16 subqueries are allowed on one side of a **union**. This does not affect most queries because only 16 tables are allowed within one query. This only affects a query with more than 16 subqueries where some of those subqueries have no **from** clauses.
- Only 40 tables are allowed in a **union all** operation.

Subqueries and NULL Results

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

are no matching rows, and raises an error. If you have applications that depend on the pre-11.x behavior, you need to rewrite them.

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

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

The correct trigger is:

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

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

No Subqueries in Updatable Cursors

Subqueries are no longer allowed in updatable cursors.

Drop and Recreate Compiled Objects Containing Subqueries

If you performed the tasks under "Tasks Specific to Rebuilding SQL Server" on page 2-11, you can skip this section. You have already created the the objects.

Stored procedures triggers, or views that contain subqueries are not automatically upgraded to take advantage of the new subquery changes. Until you drop and recreate a compiled object, the object will continue to behave as it did in earlier releases. Once you drop and recreate the object you will see the new performance and results expected from a release 11.x subquery.

➤ ***Note***

Ensure that an accurate version of your **create procedure** script exists before you drop objects.

Once the upgrade is complete, drop and recreate all compiled objects containing subqueries. You can use the new system stored procedure **sp_procqmode** to determine:

- Which objects contain subqueries
- Whether the objects are running at the release 11.x or earlier level

For details about **sp_procqmode**, see the *SQL Server Reference Manual*.

Production System Upgrade—A Word of Caution

It is essential to test compiled objects that contain subqueries.

Before Upgrading Your Production System

Test release 11.x as follows:

1. Create an adequate **test** system.

   To upgrade a test system, rename each old stored procedure and create the new release 11.x style stored procedure.

2. Analyze SQL Server 11.x behavior on the test system. Run tests to see the differences in performance and results.

3. Make the necessary changes for your application.

After Upgrading Your Production System

After you upgrade your production system:

1. Run **sp_procqmode** to determine what procedures:

   - Contain subqueries
   - Have release 10.x behavior

   Here is an example of running the command **sp_procqmode old_sproc, detail** that shows a procedure with pre-release 11 behavior:

```
Object Owner.Name  Object Type      Processing Mode  Subq  Text
-------------------------------------------------------------
dbo.au_info        stored procedure  pre-System 11   no    yes
```

   For details on **sp_procqmode**, see the *Reference Manual*.

2. Drop and recreate all compiled objects containing subqueries. Include any changes you made as a result of testing before the upgrade.

   If you run procedures with a pre-release 11.x processing mode, and then you have to drop and recreate those procedures, subqueries will exhibit SQL Server 11.x behavior.

◆ ***WARNING!***

You cannot go back to the old behavior once you drop and recreate a procedure.

3. Run user acceptance tests.

Avoid Keyword Conflicts

SQL Server 11.x provides some new Transact-SQL keywords, which you need to be aware of to avoid keyword conflicts. For example, two words you may have used as SQL identifiers are now keywords:

- **partition**
- **unpartition**

To find reserved words used as identifiers, run the new stored procedure **sp_checkreswords**, as described under "Check for Reserved Words Before Going "Live"" on page 2-9. If your existing databases use any of the new keywords as identifiers, **sp_checkreswords** captures the words in a file and generates a message like the following:

```
Warning: x conflicts with 11.0 reserved words were
found. Sybase suggests that you resolve these
conflicts before upgrading the SQL Server. Run
'sp_checkreswords' on each database for more info.
```

where *x* is the number of conflicts found.

◆ ***WARNING!***

Test all your stored procedures and triggers to be certain that they do not use keywords as identifiers. The sp_checkreswords procedure does not check the contents of stored procedures. For example, the following procedure would result in a syntax error:

```
create procedure x as
    create table user (a int)
```

Changing Object Names

Keyword conflicts yield syntax errors when objects are accessed. To avoid conflicts, perform the following tasks before migration:

1. If you have not already done so, run **sp_checkreswords** to find database objects that contain the new keywords.
2. Change the name of any database with a name that is the same as a keyword, using **sp_renamedb**.
3. Change the name of any object (table, column, and so on) that has the same name as a keyword, using **sp_rename**.
4. Remember to change any applications that reference those objects as well.

   For example, this query:

   ```
   select user from table_x
   ```

   yields a syntax error because "user" is a now keyword.

Using set Commands to Keep Object Names

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

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

For more information about:

- Descriptions of Transact-SQL keywords, see the *Reference Supplement*
- Details on changing object names, see the *SQL Server Reference Manual* under **sp_checkreswords**

Using *isql* to run *sp_checkreswords*

If you subsequently want to run reserved word checks, use the following command sequence for each database you want to check:

```
% isql -Ppassword -Sservername
1> use database_name
2> go
1> sp_checkreswords
2> go
```

Identifying New Keywords

If you want to identify the new SQL Server 11.x keywords, install **sp_checkreswords** and run the following query:

```
select name
from master..spt_values
where type = "W"
```

You can also run this query against pre-release 11.x and compare the list with the release 11.x list.

System Stored Procedures Output

Many of the existing system stored procedures provide new and improved output. If your current applications use system stored procedures, check to see what they report under SQL Server 11.x.

showplan Output

The output for **set showplan on** has been changed. Any applications that rely on the output generated by this command may need to be changed.

For details about **showplan** messages, see the *Performance and Tuning Guide*.

Improved Error Messages

Many error messages are added, and some existing messages are changed to improve their readability. If you rely on the text of any error messages within your applications, check to be sure they have not changed. You can use the following statement to see the text changes on SQL Server 11.x:

```
select * from sysmessages where error = x
```

Database Administration Changes

This section identifies a load database script change and offers tips for *sybsystemprocs* database setup.

➤ ***Note***

For RS/6000 AIX sites, also see DBA information in Appendix F, "Create Database Devices on AIX with Additional Space."

Databases Online / Offline

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

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

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

sybsystemprocs Database

Tasks related to the *sybsystemprocs* database are as follows:

- Increase the database size to make room for the new system stored procedures in release 11.x. For guidelines, see "Increase sybsystemprocs" on page 2-7.
- If you modified Sybase stored procedures, rename them **before** the upgrade.

  During upgrade, all Sybase system stored procedures are dropped from *sybsystemprocs*. If you have not renamed the modified stored procedures, the install process overwrites them and you have to re-customize them after the upgrade.

System Administration Changes

This section covers 11.x changes that may impact your system administration scripts.

New Approach for Storing Configuration Values

Prior to release 11.x, SQL Server stored configuration values in the first page of the master device, known as the "configuration block". Most of these values have been moved to a flat file known as the configuration file. Please review the following topics on backups, parameter names, script changes, and trace flags.

For details on configuring SQL Server behavior, see the *System Administration Guide*.

Configuration File Administration

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

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

sp_configure Parameter Name Changes

Some of the configuration parameters available through **sp_configure** have new names. For example:

- Memory is now called **total memory.** Running **sp_configure memory,** ***value*** results in this error: "Configuration option is not unique."
- The **allow updates** parameter name is changed to **allow updates to system tables.** To ensure that your scripts work, change any that use the syntax **sp_configure allow** to one of the following:

  ```
  sp_configure "allow updates", option
  ```

  or

  ```
  sp_configure "allow updates to system tables",
  option
  ```

If you have scripts that use **sp_configure** to set or report on configuration parameters, change them if the parameter names have changed. Be sure to test all scripts that use the parameter names listed in *What's New in Sybase SQL Server Release 11.0?*

reconfigure Command Ignored

The **reconfigure** command is no longer required after running **sp_configure.** Any of your scripts that include **reconfigure** will continue to work; the **reconfigure** command is ignored. You may want to consider removing **reconfigure** commands from your scripts now, however, to prevent problems in future releases.

Trace Flags

The following table shows the trace flags that are now configuration parameters:

Old Name	New Name
T1204	print deadlock information
T1603	allow sql server async i/o
T1610	tcp no delay
T1611	lock shared memory

Check whether you are currently using trace flags that have been converted to the configuration file and, if so, reset them after you upgrade.

If you currently set any of these trace flags in your runserver file, set them one of the following ways:

- Use **sp_configure,** or
- Edit the configuration file.

For example, if you use a trace flag to print deadlock information to your error log (**-T1204** in your runserver file on UNIX), do both of the following:

- Remove it from your runserver file.
- Set deadlock information printing with **sp_configure** or by editing your configuration file.

For details on configuration parameters, see the *System Administration Guide.*

buildmaster Flags

The **buildmaster** executable no longer supports the **-y** and **-r** flags. Use **sp_configure** or the configuration file in place of these flags. If you have scripts that use these flags, rewrite them using **sp_configure**, or save and edit copies of the configuration file.

For details about the configuration file and **sp_configure** options, see the *System Administration Guide.*

New Configuration Parameters

SQL Server 11.x includes several new configuration parameters. This section addresses only **deadlock checking period** and **page utilization percent**.

For more information:

- *What's New in Sybase SQL Server Release 11.0?* lists the new and changed parameters.
- The *System Administration Guide* details how to use the configuration parameters.

deadlock checking period Parameter

Prior to release 11.0, SQL Server initiated deadlock checks as soon as a task had to wait for a lock. To keep this behavior, set the **deadlock checking period** to 0 (zero) in the configuration file.

SQL Server 11.x performs deadlock checking after a minimum period of time for any process waiting for a lock to be released. By default, this minimum period is 500 milliseconds, but you can change this default with **deadlock checking period**.

Delaying deadlocking saves overhead. If you expect your applications to deadlock infrequently, you can reduce overhead cost by delaying deadlock checking. However, increasing **deadlock checking period** causes longer delays before deadlocks are detected.

For details about deadlock checking, see the *Performance and Tuning Guide*.

page utilization percent Parameter

The configuration parameter **page utilization percent** saves time by allocating new extents rather than searching the OAM page chain. This parameter makes page allocation faster, but it may waste space.

Prior to release 11.0, SQL Server searched the OAM page chain for unused pages before allocating a new extent. To keep this behavior, use **sp_configure** to set **page utilization percent** to 100.

Memory and Cache

Release 11.x uses more memory for the Sybase kernel and for internal structures, including the new user log cache. You may need to add

more total memory to your server to maintain the same performance as your previous release.

Also, compiled objects grew about 25% in release 11.x. You may need to enlarge your procedure cache to maintain the same performance. Use the following rule of thumb to estimate the memory increase needed:

For	Increase Memory
Kernel and internal structures	5MB (could be more for multi-engine)
User connections	15K
Stored procedures	25%

➤ ***Note***

Named data caches allow you to use memory more effectively, which requires memory in addition to these rule-of-thumb estimates. For more information, see the *SQL Server Performance and Tuning Guide*.

Reconfigure Memory for Normal Usage

After the upgrade to release 11.x, follow these steps to return your memory parameters to their normal configurations. Use the record of memory usage you created as described in "Setting Up Memory and Cache for an Upgrade" on page 2-6.

1. Save the current settings. You can find out what the current values are using **sp_configure.**

1. Check the error log to see how much memory is left in cache. By subtracting this number from the amount you recorded earlier, you can determine how much additional memory is required by the SQL Server.

2. From the baseline information you recorded earlier, use **sp_configure** to reconfigure the **total memory** parameter as needed to support your usual number of **user connections**, **open objects**, and **locks**.

➤ ***Note***

You may need to add more memory for procedure cache because stored procedures now need about 25 percent more memory.

3. After making the changes, check the error log again and record how much cache you have now.

   Be sure that the default data cache has the same size as your current system's data cache after all other allocations. Otherwise, performance may degrade, depending on the application.

For more information about:

- Configuration settings, see the SQL Server configuration documentation for your platform
- Using **sp_configure**, see the *System Administration Guide*

Backup Server

The Backup Server has a feature to deal with unfamiliar tape devices. If you dump a database to a tape device that is not one of the devices mentioned in the *System Administration Guide* for your platform, and Backup Server cannot determine the device type, the **dump** command fails.

Consequently, when you first dump to a new tape device, use the **with init** option to the **dump** command. The Backup Server first reads and writes to the tape in order to determine how to communicate with it. When Backup Server finishes this first test, it writes a line to the new tape configuration file, called *$SYBASE/backup-tape.cfg* by default, which is created during upgrade or install.

➤ ***Note***

Manage the *backup-tape.cfg* file as part of the backup strategy for your system.

Preview Performance

This section assumes you have completed installing and configuring SQL Server and Backup Server.

Execute Tests

Execute both your application test suite to validate results and performance scripts to measure performance goals.

If you are using only one machine to migrate from one system to another and you are doing the migration and validation on a weekend or during off hours:

- Arrange to have weekend or off hours support from Sybase, as well as the operating system and hardware vendors.
- Consider mirroring SQL Server in addition to having a pre-migration backup. For more information about mirroring disk devices, see the Sy*stem Administration Guide*.

Compare Release 11.x Tests to Release 10.x Baseline

Once you have established a release 10.x baseline as described in Chapter 1, "Planning Migration," create a release 11.x test system. Run test scripts, and compare the outputs between the two releases.

The steps include:

1. Upgrade the test server to release 11.x, as described in this chapter. This important step allows you to catch and resolve problems before doing the production system upgrade.
2. Perform database consistency checking on all release 11.x test databases, including *master* and *sybsystemprocs*. Run **dbcc** with options **checkdb**, **checkalloc**, and **checkcatalog**.
3. Back up the databases.
4. Run single-user tests on the release 11.x test system to validate optimizer decisions as follows:
   - Capture **showplan**, **statistics io**, **statistics time**, and **dbcc traceon(302,310)** output.
   - Compare the output to that of the release 10.x test system. Look for the same or better optimizer decisions, and the same logical and physical I/O counts.
   - Fix all query plan problems now. Otherwise, they may be difficult to distinguish from other problems that surface later. Unresolved query problems also can invalidate multi-user tests, requiring extra problem-solving and testing time.
5. Iteratively run multi-user tests for each component as follows:
   - Capture the transaction throughput and response metrics.

- Compare the throughput and metrics to those of the release 10.x test and productions systems.
- Resolve obvious problems.
- Note opportunities for tuning. For information about the various layers for tuning, see the *Performance and Tuning Guide*.
- To verify that there is no data corruption, periodically run **dbcc** with options **checkdb**, **checkalloc**, and **checkcatalog**. This is not necessary after every test.
- Restore the databases to a known state, reloading them from backup.
- Re-run the tests until you are satisfied.

6. Back up the release 11.x test system databases.

Run Followup Tests

Execute the following tests against the release 11.x test system:

- Integration tests, to make sure that all the system components work together
- End-user acceptance tests to verify that user standards are satisfied
- Upgrade tests and fallback plan tests to make sure no steps are missing and contingencies are covered

For more information about these tests, see Chapter 1, "Planning Migration."

Performance Tuning Tips

The *Performance and Tuning Guide* provides in-depth system administration, database design, and application programming information.

To immediately enhance SQL Server 11.x functions, such as subqueries, **bcp** function, dumps, **dbcc** commands, **create index** statements, and **select** statements with large result sets, perform these tuning actions:

- **Optimizer strategy**—Determine whether the optimizer is choosing the new MRU strategy correctly. See the *Performance and Tuning Guide* for information about:

  - The **forceplan** option and specifying the index for a query
  - The **sp_cachestrategy** procedure for specifying cache strategy
- **Index choice**—Make sure you have the right set of indexes on your tables, so that queries access data with a minimum number of page reads. For details on indexing, see the *Performance and Tuning Guide*.
- **Memory/cache allocation**—Increase SQL Server 11.x memory to break even with that of your current system. Do not set memory too low. See "Memory and Cache" on page 2-24.
- **Performance monitoring**—Use **sp_sysmon**, the stored procedure that produces SQL Server performance data that you can use for tuning. For details on monitoring performance with **sp_sysmon**, see the *Performance and Tuning Guide*.

If performance drops after upgrading from release 10.x to 11.x, research these options:

- Put subquery changes into effect. See "Drop and Recreate Compiled Objects Containing Subqueries" on page 2-16.
- Prevent the housekeeper task from increasing database writes. Set **housekeeper free write percent** to 0 (zero). See the *System Administration Guide*.

Practice!

Explore changes from release 10.x, such as:

Changed or New	For more information
Named data caches	*Performance and Tuning Guide*
Large I/O	*Performance and Tuning Guide*
Query tuning options	*Performance and Tuning Guide* *Analyzing, Handling, and Resolving Optimizer Problems/Symptoms* (in the Technical Information Library on the Internet)
Configuration file and **sp_configure**	*System Administration Guide*
sp_sysmon to monitor counters and analyze performance	*Performance and Tuning Guide*

Changed or New	For more information
Reorganized documentation for:	See the following, respectively:
• System table information	• *Reference Supplement*
• Security information	• *Security Administration Guide*

For a more detailed list of changes, see *What's New in Sybase SQL Server 11.0?*

3 Migrating from Release 4.x to 11.x

Read this chapter if you are migrating to SQL Server 11.x from SQL Server release 4.x.

For release 10.x migration, see Chapter 2, “Migrating from Release 10.x to 11.x.”

This chapter covers the application and administrative changes that are critical to migration from release 4.x, as well as other considerations and recommended testing. The topics include:

➤ ***Note***

For Microsoft SQL Server users, *Installing Sybase SQL Server for Windows NT* tells you how to upgrade a Microsoft SQL Server 4.2 to Sybase SQL Server 11.x. In addition, follow the guidelines in this chapter.

Upgraded Server versus Rebuilt Server

For migration to SQL Server 11.x, you perform either an upgrade or rebuild process:

Table 3-1: SQL Server upgrade versus rebuild

Process	Purpose	How	Advantages	Disadvantages
Upgrade	"Promote" databases from release 4.x to 11.x.	Use the install/configure utility for your platform: • **sybinit** (UNIX) • **Server Config** (Windows NT) For more information, see installation and configuration instructions for your platform.	Simple to implement. Low risk.	Limited flexibility for modifying environment during the upgrade.
Rebuild	Construct a release 11.x environment, then apply your release 4.x environment.	Use one of the following: • Locally developed scripts or programs. • **bcp** and **defncopy** utilities. **Note: defncopy** supports defaults, rules, triggers, views, and stored procedures, but not tables, indexes, constraints or declarative referential integrity.	An opportunity to improve system design. More flexibility for environment modifications.	Resource intensive, including time, hardware, staff. Potentially complex and thus can introduce risk.

Prepare for Migration

This section assumes you have done the necessary planning, as outlined in Chapter 1, "Planning Migration," and you have installed the necessary hardware and software, such as Replication Server.

Summary of Steps

The following is a summary of the SQL Server installation steps. Complete these steps, detailed in the installation instructions for your platform, using the information in this chapter where indicated.

Table 3-2: Summary of installation steps

Phase	Tasks
Planning	1. Read the SQL Server Release Bulletin.
	2. Create a "sybase" account to perform installation and configuration tasks.
	3. Locate your Customer Authorization String (CAS) on the CD-ROM or tape package.

Table 3-2: Summary of installation steps (continued)

Phase	Tasks
Preparing for Installation	Choose to install a new SQL Server 11.x if you are following a rebuild plan. You may also decide to install a new 11.x server so you can test it. The following is the sequence of tasks to prepare for install. For details, refer to the installation instructions for your platform. 1. Create the Sybase installation directory. 2. Set the $SYBASE environment variable to the path of the Sybase installation directory. 3. Verify permissions on your Sybase installation directory. 4. Determine where to install the new release. 5. Back up the existing Sybase installation directory. 6. If installing other Sybase products, determine which ones should be running. 7. Consider product version and compatibility issues. 8. Make sure that your operating system supports the products you want to install. Also see “Prepare Platform” on page 3-7. 9. Make sure that your operating system configuration and available disk space are adequate to complete the installation. Also see “Setting Up Memory and Cache for an Upgrade” on page 3-7. 10. Install operating system kernel patches. Also see “Apply Prerequisite Platform Fixes” on page 3-6. 11. Verify your network software configuration. 12. If you are installing remotely from tape, be sure that access permissions are configured correctly. 13. Be sure the $DISPLAY environment variable is set to the computer on which you are running **sybsetup**. 14. Make sure your operating system configuration is adequate to run SQL Server. 15. Set shared memory parameters. 16. For some UNIX platforms, enable asynchronous disk I/O. 17. Edit the default login file, if needed. 18. Choose database devices.
Installing	1. Complete the Configuration and Upgrade Worksheet in the installation guide for your platform. 2. Follow the step-by-step installation instructions for your platform.

Table 3-2: Summary of installation steps (continued)

Phase	Tasks
Preparing for the Upgrade	These upgrade tasks apply if you are following a plan to upgrade your server in place, or if you upgrade an existing test server. For complete details on these steps, refer to the installation instructions for your platform. 1. Before upgrading a SQL Server that contains replicated databases, follow the instructions for a replicated system in your installation guide. 2. Make sure that you installed the 11.x software in a different directory from your previous SQL Server installation. 3. Identify the name and location of the runserver file. 4. If you have not already done so, record your configuration. The installation guide provides a worksheet. 5. Test and change your current applications and stored procedures, as needed. • For guidelines on testing see Chapter 1, “Planning Migration.” • For information on potential release 11.x impact on your applications, see “Application Changes” on page 3-13. 6. Determine whether your current SQL Server supports upgrade to release 11.x. • You can upgrade Sybase SQL Server 4.9.2 and higher. • You can upgrade Microsoft SQL Server 4.2. Follow the special instructions in the installation guide for Windows NT platforms. 7. Find space for the new *sybsystemprocs* database. See “New Database for Stored Procedures” on page 3-8. 8. Check for reserved word conflicts. Also see “Check for Reserved Words Before Going “Live”” on page 3-10. 9. Make sure that all SQL Server users are logged off. Also see “Log Off Users” on page 3-11. 10. Check database integrity. Also see “Run dbcc Options” on page 3-11. 11. Back up all databases on your current SQL Server. 12. Turn off databases options. “Turn Off Options for All Databases but tempdb” on page 3-9. 13. Configure memory resources. 14. Note the current cache size.
Upgrading	Perform the actual upgrade. See the following information: • Step-by-step instructions in the installation guide for your platform • “Run sybinit (UNIX) or Server Config (Windows NT)” on page 3-11 **Note:** Remember to install Backup Server.

Table 3-2: Summary of installation steps (continued)

Phase	Tasks
Followup	1. Verify your new SQL Server version. 2. Reset configuration options. “New Approach for Storing Configuration Values” on page 3-34. 3. Reset database options. 4. Update your scripts. For information on potential release 11.x impact on your scripts, see “Database Administration Changes” on page 3-32 and “System Administration Changes” on page 3-33. 5. Increase the procedure cache. Also see “Queries” on page 3-13. 6. Mirror devices, if desired. 7. Create last chance threshold procedure. Also see “Decide Last Chance Threshold Behavior” on page 3-38. 8. Drop and recreate procedures with subqueries. Also see “Queries” on page 3-13 and “Drop and Re-create Compiled Objects Containing Subqueries” on page 3-21.

Prepare the Environment

The topics in this subsection highlight a few tasks in the SQL Server installation instructions. Before performing them, be sure that you have already completed both of the following:

- Instructions on preparing for a new installation or an upgrade
- The worksheet in the installation guide for your platform

Apply Prerequisite Platform Fixes

Apply prerequisite platform / operating system fixes, based on the *Release Bulletin* and support structure for the platform. If applicable, do the following:

- For an upgrade, test the SQL Server 4.x with any new operating system patches. If the test is successful, you will be able to run release 4.x and 11.x on the same operating system.
- Avoid changing the operating system and SQL Server at the same time, unless you are building a new system. Instead, upgrade the operating system first, stabilize it, and then upgrade SQL Server.

Prepare Platform

Make platform-specific preparations as follows:

Platform	Action Before Running *sybinit* or *Server Config*
RS6000 / AIX	To install or upgrade SQL Server on RS6000 / AIX, do the following: • Allow 1MB for Logical Volume Control Block. For more information, see Appendix F, "Create Database Devices on AIX with Additional Space." • For an install only, check the System Management Interface Tool (SMIT) asynchronous input / output parameters. For more information, see SQL Server installation guidelines and your operating system documentation.
SunOS	Before upgrading a SQL Server currently on SunOS, migrate to Solaris according to the release 11.x installation instructions. Also see Appendix D, "FAQs: Migrating SQL Server 4.x or 10.x on SunOS to 11.x on Solaris."

Setting Up Memory and Cache for an Upgrade

Changes in SQL Server 11.x have increased its memory requirements, such as:

- New user log cache
- Larger **dataserver** binary
- Previously existing structures which are now larger (such as locks)

To avoid upgrade problems based on limited memory, perform both of the following steps before the upgrade:

1. **Record the current cache sizes.**

   Look at the error log to get a profile of the memory usage on your production SQL Server, including the buffer cache, procedure cache, and procedure header sizes.

   The following is an example of memory-related messages from the error log for SQL Server with **total memory** parameter of 7500:

```
server: Number of proc buffers allocated: 556
server: Number of blocks left for proc headers: 629
server: Memory allocated for the default data cache: 4144 Kb
```

For an upgrade to release 11.x, you set memory requirements for the upgrade itself, and then use these recorded values as a baseline for determining requirements after the upgrade.

For more information about configuring memory and memory-related messages in the error log, see the *System Administration Guide*.

2. **Configure memory resources.**

   Because of extra overhead requirements, the upgrade process can run out of memory if you do not increase it. You can increase memory in one of two ways:

   - Raise the amount of allocated memory by changing the value of the **total memory** configuration parameter.
   - Lower certain parameters, such as **user connections**, to as close to the default values as possible.

     The defaults for some configuration parameters that consume memory are:

Configuration Parameter	Default Value
user connections	25
locks	5000
open objects	500

For post-upgrade considerations, see “Increased Memory and Cache Needs” on page 3-37.

Tasks Specific to Upgrading SQL Server

If you are upgrading SQL Server 4.x to SQL Server 11.x, perform the tasks in this section.

If you are rebuilding SQL Server, see “Tasks Specific to Rebuilding SQL Server” on page 3-12.

New Database for Stored Procedures

System stored procedures are stored in a new database called *sybsystemprocs*. You may need to modify your **dbcc**, backup and recovery procedures to include this database.

Configure Enough Devices to Add *sybsystemprocs*

For an upgrade, you need to find space for *sybsystemprocs* on an existing or a new device.

Check that you have enough SQL Server devices configured to create the *sybsystemprocs* device. For example if you have 11 devices configured, and all of them are already in use, increase devices by 1 as follows:

```
sp_configure devices, 12
go
reconfigure
go
```

Reboot SQL Server.

Disable Replication

Disable replication and clear the log before upgrading to release 11.x. If replication is enabled and your log has not been cleared, the upgrade will fail. For more information:

- The installation guide for your platform details how to handle the Replication Server with a SQL Server upgrade, including the steps necessary to disable replication.
- The *Replication Server Commands Reference* details commands.
- The *System Administration Guide* details how to clear logs.

Turn Off Options for All Databases but *tempdb*

Turn off all options on all databases, except *tempdb*, using **sp_dboption**. The upgrade requires that you set **select into/bulk copy** to true for *tempdb*.

➤ ***Note***

This is a good time to reconsider your settings for database options.

To find out what options are set, use the **sp_helpdb** stored procedure.

To turn off an option, use this syntax:

```
sp_dboption database_name, "option", false
go
use database_name
go
checkpoint
go
```

To set **select into/bulk copy** for *tempdb*, use the following command:

```
sp_dboption tempdb, "select into/bulkcopy", true
go
use tempdb
go
checkpoint
go
```

See the *SQL Server Reference Manual* for information on **sp_dboption**.

Check for Reserved Words Before Going "Live"

We strongly recommend that you check for the use of reserved words before you upgrade a **production** system. Check for reserved words when you test applications and resolve the conflicts at that time. For details, see "Avoid Keyword Conflicts" on page 3-23.

Tthe **sp_checkreswords** stored procedure checks for reserved word conflicts. Change database names that conflict with reserved words immediately. Otherwise, the upgrade will fail.

To find reserved word conflicts, run the **sp_checkreswords** stored procedure as follows:

Platform Type	To Run *sp_checkreswords*	For More Information
Windows NT	Server Config installs **sp_checkreswords** and performs reserved word checking during upgrade.	*Installing Sybase SQL Server for Windows NT* *Reference Manual* under **sp_checkreswords**
UNIX	**sybinit** installs **sp_checkreswords** automatically. You can use **sybinit** to run **sp_checkreswords** at any time, including before an upgrade, as follows: 1. Begin a **sybinit** session, as described in the installation guide. 2. Select "Check for reserved word conflicts" from the SQL Server Upgrade menu.	*SQL Server Installation and Configuration Guide* for your UNIX platform *Reference Manual* under **sp_checkreswords**

Log Off Users

Immediately before proceeding with the migration, be sure that:

- SQL Server is running.
- All users are logged off and applications that connect to SQL Server are not running.

  Use **sp_who** to determine what users and processes are online.

Run *dbcc* Options

To ensure healthy databases, run the following **dbcc** options on all pre-release 11.x databases:

- **checkdb**
- **checkalloc**
- **checkcatalog**

See the *System Administration Guide* for instructions on running **dbcc**.

Run *sybinit* (UNIX) or *Server Config* (Windows NT)

For instructions on using **sybinit** or **Server Config**, see the SQL Server installation instructions for your platform.

Additional points are:

- Make sure that *sybsystemprocs* size accommodates the system procedure **sp_sysmon** and the system procedures that support **sp_thresholdaction**.
- For a log of the upgrade process:

Platform	Install/Upgrade Log Location	How Created
Windows NT	*<install_path> \ init \ logs \ log <mmdd>*.xxx	The installation process automatically creates the log.
UNIX	*$SYBASE/init/logs/log <mmdd>*.xxx	Type Ctrl- w at the end of a **sybinit** session to retrieve a file of all the selections that you made during the session.

➤ ***Note***

If your upgrade attempt is unsuccessful, see Appendix C, "Recovery from Upgrade Failure."

Tasks Specific to Rebuilding SQL Server

If you are building a SQL Server 11.x environment, perform the following tasks. This section assumes that you have already performed the tasks under "Prepare for Migration" on page 3-3.

Before You Begin

Run **sybinit** (UNIX) or **Server Config** (Windows NT) to install a SQL Server, as described in the SQL Server installation documentation for your platform.

Check for Reserved Words

You can check for reserved words in either of the following ways:

- Run **sybinit** or **Server Config** against your pre-release 11.x SQL Server, as described in "Check for Reserved Words Before Going "Live"" on page 3-10.
- Use **isql** to run the **installupgrade** script found in the release 11.x *$SYBASE* directory under *scripts*.

Create Databases

1. Use existing scripts, SQL Server Manager, or third party tools to create:
   - Database objects
   - User stored procedures
   - Triggers
   - Views
   - Users
   - Permissions

You can also **defncopy** to create user stored procedures, triggers, and views.

2. Move the data using the **bcp** utility.

   bcp is much faster than it was in release 10.x. Partitioning your target tables, combined with the use of multiple **bcp** streams, makes moving data even faster.

   For more information about:

   - Using **bcp**, see the utilities manual for your environment
   - Partitioning tables, see *Performance and Tuning*

Application Changes

Before upgrading your **production** system, review this section. Consider testing to highlight potential problems.

This section describes ANSI standardization and changes to queries and keywords that may impact your applications. It also notes features that may be visible through applications after migration.

➤ ***Note***

For a summary of ANSI SQL '89 and FIPS 127-1 support, see Appendix H, "ANSI SQL '89 and FIPS 127-1 Standards Compliance."

Queries

Changes to subquery processing, the optimizer, and subquery performance testing.

Subquery Processing

Changes to subquery processing support full ANSI compatibility and fix some bugs in SQL Server. These changes may show different results. The changes, detailed below, affect the following predicates:

- **in** / **any**
- **not in**
- **or** with **exists**, **in**, or **any**
- **>all** / **<all**

- **exists** (aggregates)
- **distinct** (**select**)
- **between**

Be sure to test subqueries used in your applications to understand the new behavior. You may need to rewrite your subqueries to gain the benefits of correct and smaller result sets for these predicates.

in/any

Subqueries that use **in/any** no longer return duplicates. In release 4.x duplicate rows returned from the outer table were equal to the number of rows from the inner query for each outer row.

Benefits of the new behavior are:

- Smaller results sets
- Reduced application code in ignoring duplicates
- ANSI compatibility

Change applications that require release 4.x behavior so they do not break.

The following example compares the results for each release using the *pubs* database:

Query	Release 4.x Results	Release 10.x and 11.x Results
`select pub_name from publishers where pub_id in (select pub_id from titles)`	New Age Books New Age Books New Age Books New Age Books New Age Books Binnet & Hardley Binnet & Hardley Binnet & Hardley Binnet & Hardley Binnet & Hardley Binnet & Hardley Binnet & Hardley Algodata Infosystems Algodata Infosystems Algodata Infosystems Algodata Infosystems Algodata Infosystems Algodata Infosystems	New Age Books Binnet & Hardley Algodata Infosystems

not in

As of release 10.x, when subqueries that use **not in** return NULL, ANSI semantics process them as FALSE. release 4.x returned TRUE if these subquery results contained no matching values but did contain NULL.

This example shows how SQL Server releases evaluate a query when pub_id 0877 and 1389 contain all nulls in the price column:

Query	Release 4.x Results	Release 10.x and 11.x Results

`select pub_id`
`from publishers`
`where $100.00 not in`
`(select price from titles`
`where titles.pub_id=`
`publishers.pub_id)`
	0736
	0877
	1389
		0736

or...exists/in/any

Subqueries that contain **exists**, **in**, or **any** under an **or** return correct results sets. release 4.x prevented rows from being returned if subqueries evaluated to FALSE, even if the **or** clause was TRUE.

This example shows how SQL Server releases evaluate a query that contains **in** and a true **or** clause:

Query	Release 4.x Results	Release 10.x and 11.x Results

`select pub_name`
`from publishers`
`where pub_id in`
`(select pub_id from titles`
`where title = “No Such Book”)`
`or pub_id = ‘1389’`
	NULL	Algodata Infosystems

>all and <all

Subqueries that contain **>all** and **<all** return the correct results. ANSI standards state that these subqueries are TRUE when they returns no rows.

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

Query	Release 4.x Results	Release 10.x and 11.x Results

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

For examples and explanations of using the **all** keyword, see the *Transact-SQL User's Guide*.

Correlated Subqueries

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

If you rewrite release 4.x subqueries for release 11.x, they will return the correct results. Impacts of the new behavior are:

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

Here is an example that compares releases, using the *pubs* database:

Query	Release 4.x Results	Release 10.x and 11.x Results

`select pub_id,`
`(select count(*) from`
`publishers`
`where publishers.pub_id =`
`titles.pub_id`
	0736 5
	0877 7
	1389 6
		0736 5
		0736 5
		0736 5
		0736 5
		0736 5
		0877 7
		0877 7
		0877 7
		0877 7
		0877 7
		0877 7
		0877 7
		1389 6
		1389 6
		1389 6
		1389 6
		1389 6
		1389 6

Aggregates with exists

In release 4.x SQL Server, queries that had both aggregates and **exists** subqueries sometimes returned the wrong answer. This happened for a correlated subquery where a table in the subquery's **from** clause had duplicates.

This example compares release results when an **exists** subquery uses a **from** clause against the *pubs2* database:

Query	Release 4.x Results	Release 10.x and 11.x Results

`select count(*)`
`from publishers`
`where exists`
`(select * from titles`
`where titles.pub_id =`
`publishers.pub_id)`
	18	3

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

select distinct

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

This example compares releases, using the *pubs* database:

Query	Release 4.x Results	Release 10.x and 11.x Results

`select pub_name`
`from publishers`
`where pub_id in`
`(select distinct pub_id from`
`titles`
`where titles.pub_id =`
`publishers.pub_id)`
	NULL	New Age Books
		Binnet & Hardley
		Algodata Infosystems

between

ANSI standards state that the following predicates are equivalent:

```
expr1 between expr2 and expr3
expr1 >= expr2 and expr1 <= expr3
```

The release 4.x SQL Server switches *expr2* and *expr3* automatically if it knows that *expr2* > *expr3* at compile time. In release 10.x and 11.x, SQL Server no longer does that switch.

Here is an example that compares releases:

Query	Release 4.x Results	Release 10.x and 11.x Results

`create table demo (id int)`
`insert into demo values (250)`
`select id from demo`
`where id between 400 and 200`
	250	no rows returned

New Most-Recently-Used Plan

A new strategy known as **MRU** (most recently used or fetch-and-discard) is available to the optimizer. The MRU strategy is designed to reuse the same buffers in cache and to avoid overwriting all existing pages.

For example, on SQL Server releases prior to 11.x, users running small transactions often find the necessary pages in cache (avoiding

physical I/O). When another user runs a **select *** query from a very large table, the cache is overwritten with pages from this table. This forces the other users' queries to use more physical I/O because their pages are no longer in cache.

The release 11.x optimizer can choose either LRU (least recently used) or MRU. To see your optimizer's plan, execute **set showplan on** and then run your query. If you see LRU, your SQL Server is running with old behavior; if you see MRU, it is running with the new 11.x behavior.

You can override the MRU plan if you are unhappy with its performance by using the l**ru** option in queries, described in the section on **select** in the *SQL Server Reference Manual*. To get pre-11.x behavior for your queries, construct it as follows:

```
select * from table (index table prefetch 2 lru)
```

For details on advanced optimizer techniques, see the *Performance and Tuning Guide*.

Subquery Performance

If you notice performance degradation after upgrade, check your data cache and increase it if necessary. SQL Server 11.x takes memory from the data cache if it does not have enough, causing the cache to shrink. For more information, see "Increased Memory and Cache Needs" on page 3-37.

Expression Subqueries

The SQL Server 11.x optimizer does not flatten a query where:

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

Such a query might look like this:

```
select * from huge_table where x=
    (select sum(a) from tiny_table
    where b = huge_table.y)
```

If you find this type of query to be slow, you can reformulate the query, as follows:

```
select huge_table.y, s=sum(a)
    into #t
    from huge_table, tiny_table
    where b=huge_table.y
    group by huge_table.y

select huge_table.*
    from huge_table, #t
    where x=#t.s
    and huge_table.y=#t.y
```

union Limitations

The following limits apply to the **union** operator:

- Only 16 subqueries are allowed on one side of a **union**. This does not affect most queries because only 16 tables are allowed within one query. This only affects a query with more than 16 subqueries where some of those subqueries have no **from** clauses.
- Only 40 tables are allowed in a **union all** operation.

Subqueries and NULL Results

Prior to SQL Server 11.x, a correlated expression subquery in the **set** clause of an **update** returned 0 instead of NULL when there were no matching rows. SQL Server 11.x correctly returns NULL when there are no matching rows, and raises an error. If you have applications that depend on the pre-11.x behavior, you need to rewrite them.

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

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

The correct statement is:

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

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

Drop and Re-create Compiled Objects Containing Subqueries

If you performed the tasks under "Tasks Specific to Rebuilding SQL Server" on page 3-12, you can skip this section. You have already created the the objects.

Stored procedures, triggers, or views that contain subqueries are not automatically upgraded to take advantage of the new subquery changes. Until you drop and re-create a compiled object, the object will continue to behave as it did in earlier releases. Once you drop and re-create the object you will see the new performance and results expected from a release 11.x subquery.

➤ ***Note***

Ensure that an accurate version of your **create procedure** script exists before you drop objects.

Once the upgrade is complete, drop and re-create all compiled objects containing subqueries. You can use the new system stored procedure **sp_procqmode** to determine:

- Which objects contain subqueries
- Whether the objects are running at the release 11.x or earlier level

For details about **sp_procqmode**, see the *SQL Server Reference Manual*.

Production System Upgrade—A Word of Caution

It is essential to test compiled objects that contain subqueries.

Before Upgrading Your Production System

Test release 11.x as follows:

1. Create an adequate **test** system.

   To upgrade a **test** system, rename each old stored procedure, create the new release 11.x style stored procedure.
2. Analyze SQL Server 11.x behavior on the test system. Run tests to see the differences in performance and results.
3. Make the necessary changes for your application.

After Upgrading Your Production System

After you upgrade your production system:

1. Run **sp_procqmode** to determine what procedures:
   - Contain subqueries
   - Have release 10.x behavior

   Here is an example of running the command **sp_procqmode old_sproc, detail** that shows a procedure with pre-release 11 behavior:

```
Object Owner.Name  Object Type      Processing Mode  Subq  Text
-----------------------------------------------------------------
dbo.au_info        stored procedure pre-System 11    no    yes
```

   For details on **sp_procqmode**, see the *Reference Manual*.

2. Drop and recreate all compiled objects containing subqueries. Include any changes you made as a result of testing before the upgrade.

   If you run procedures with a pre-release 11.x processing mode, and then you have to drop and recreate those procedures, subqueries will exhibit SQL Server 11.x behavior.

◆ ***WARNING!***

You cannot go back to the old behavior once you drop and recreate a procedure.

3. Run user acceptance tests.

Avoid Keyword Conflicts

SQL Server 11.x provides some new Transact-SQL keywords, which you need to be aware of to avoid keyword conflicts. For example, two words you may have used as SQL identifiers are now keywords:

- **partition**
- **unpartition**

To find reserved words used as identifiers, run the new stored procedure **sp_checkreswords**, as described under "Check for Reserved Words Before Going "Live"" on page 3-10. If your existing databases use any of the new keywords as identifiers, **sp_checkreswords** captures the words in a file and generates a message like the following:

```
Warning: x conflicts with 11.0 reserved words were
found. Sybase suggests that you resolve these
conflicts before upgrading the SQL Server. Run
'sp_checkreswords' on each database for more info.
```

where *x* is the number of conflicts found.

◆ ***WARNING!***

Test all yourstored procedures and triggers to be certain that they do not use eywords as identifiers. The sp_checkreswords procedure does not check the contents of stored procedures. For example, the following procedure would result in a syntax error:

```
create procedure x as
    create table user (a int)
```

Changing Object Names

Keyword conflicts yield syntax errors when objects are accessed. To avoid conflict, perform the following tasks before migration:

1. If you have not already done so, run **sp_checkreswords** to find database objects that contain the new keywords.
2. Change the name of any database with a name that is the same as a keyword, using **sp_renamedb**.
3. Change the name of any object (table, column, and so on) that has the same name as a keyword, using **sp_rename**.
4. Remember to change any applications that reference those objects as well.

   For example, this query:

   ```
   select user from table_x
   ```

   yields a syntax error because "user" is a now keyword.

Using set Commands to Keep Object Names

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

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

For more information about:

- Descriptions of Transact-SQL keywords, see the *Reference Supplement*
- Details on changing object names, see the *SQL Server Reference Manual* under **sp_checkreswords**

Using *isql* to run *sp_checkreswords*

If you subsequently want to run reserved word checks, use the following command sequence for each database you want to check:

```
% isql -Ppassword -Sservername
1> use database_name
2> go
1> sp_checkreswords
2> go
```

Identifying New Keywords

If you want to identify the new SQL Server 11.x keywords, install **sp_checkreswords** and run the following query:

```
select name
from master..spt_values
where type = "W"
```

You can also run this query against pre-release 11.x and compare the list with the release 11.x list.

Semantic Changes

For ANSI compliance, release 10.0 enacted changes to comments and Transact-SQL syntax, which release 11.0 contains.

Comment Protocol

ANSI comments are started with two or more consecutive hyphens (--) and are terminated by a new line (<CR><LF>).

ANSI comments co-exist with the Transact-SQL® comments as of the release 10.x and 11.x SQL Server.

Certain mathematical expressions return different results in the release 10.x and 11.x SQL Server because of the support of ANSI comments.

For example, for the following query:

```
select 5--2
```

- Release 4.x returns 7.
- Release 10.x and 11.x returns 5, because "--2" is considered a comment.

To get back the value 7 under SQL Server 10.x and 11.x, use either:

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

or

```
select 5 - -2
```

Transact-SQL Syntax

In release 4.x SQL Servers the following syntax was allowed:

```
select * from table1, table1
where clause
```

ANSI standards state this syntax is invalid; therefore this query returns a syntax error in release 10.x and 11.x. The correct syntax in release 10.x and 11.x is:

```
select * from table1 t1, table1 t2
where clause
```

The following update statement also returns a syntax error in release 10.x and 11.x:

```
update table1
set a = a + 1
from table1
where b = 5
```

The correct syntax is:

```
update table1
set a = a + 1
from table1 t1
where t1.b=5
```

No Null Column Headings

Previous SQL Server releases allowed NULL column headings in tables created using **select into**. ANSI standards do not allow this. As of release 10.x and 11.x, you must provide a column heading that is a valid SQL identifier. Check all applications that use **select into**.

Examples of select list items that require column headings are:

- An aggregate function, such as **avg(advance)**
- An arithmetic expression, such as **colname * 2**
- String concatenation, such as **au_lname + ", " + au_fname**
- A built-in function, such as **substring(au_lname,1,5)**
- A constant, such as **"Result"**

To specify column headings, you can use one of three different syntaxes. For example:

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

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

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

Correlation Name Consistency

As of release 10.x and 11.x, for ANSI compliance, self-joins require correlation names. Statements that specify correlation names must use them consistently, in accordance with ANSI. For example, the following query would return errors:

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

The correct query is:

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

When a subquery includes the following kind of correlation, no error is reported, but queries may return different results than a release 4.x Server:

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

In release 4.x, *mytable.columnC* in the above subquery referred to the *mytable* in the subquery. In release 10.x and 11.x, *mytable.columnC* refers to the outer table *mytable*.

If the query needs to refer to *mytable* in the subquery, construct it as follows:

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

See the *Transact-SQL User's Guide* for information about self-joins and subqueries with correlation names.

isql Display Format

The **isql** format for approximate numeric datatypes displays additional digits of precision as follows:

- Maximum units of precision for storage and display are machine dependent.
- *real* values display up to 9 digits of precision.
- *float* values display up to 17 digits.
- Values are rounded to the last digit on display. Previously, only six places to the right of the decimal were displayed.

All values requiring more digits than the maximum are displayed in scientific notation, that is, a *float* value "1e18" is displayed as such, rather than 1000000000000000000.000000. Note that the new exact numeric types, *decimal* and *numeric*, display the entire number.

Datatypes

This section covers datatype changes.

New Numeric Datatype

SQL Server 10.x and 11.x provides a *numeric* datatype. Unlike *float*, it is platform independent and exact. For a constant such as:

```
5.1
```

SQL Server 10.x and 11.x assigns the *numeric* datatype rather than *float*.For this example, if you want *float*, represent the constant as:

`5.1e0`

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

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

The migration impact includes:

- Benefits
  - An exact numeric result accommodates user-defined ranges and storage sizes.
  - There is less risk from loss of scale due to a platform's *float/real* implementation.
- Risks
  - Applications may break because functions return different datatypes than before. Pre-release 10.x Client-Library applications map *numeric* to *float*.
  - Poor query plans can prevent optimization of some clauses because *float* columns are joined to *numeric* arguments.

➤ ***Note***

If you run a release 4.x front end with SQL Server 10.x and 11.x, the *numeric* datatype value is mapped to *float* on the front end.

Online Datatype Hierarchy

The datatype hierarchy determines the datatype of computational results. The result value is assigned the datatype of its parent components closest to the top of the hierarchy. There is no loss of precision due to conversions. However, SQL Server may return different datatypes than before to applications.

For release 4.x, the *money* datatype was higher than *float* datatypes. For ANSI compatibility, release 10.x moves *money* below the *float* and *numeric* datatypes.

To determine the SQL Server datatype hierarchy use the following query:

```
select name, hierarchy
from systypes
order by hierarchy
```

The result of this query is:

Name	Hierarchy
floatn	1
float	2
datetimn	3
datetime	4
real	5
numericn	6
numeric	7
decimaln	8
decimal	9
moneyn	10
money	11
smallmoney	12
smalldatetime	13
intn	14
int	15
smallint	16
tinyint	17
bit	18
varchar	19
sysname	19
nvarchar	19
char	20
nchar	20

Name	Hierarchy
varbinary	21
timestamp	21
binary	22
text	23
image	24

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

```
select $12*8.9
```

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

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

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

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

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

Conversions

Changes to datatype conversion include:

- **Numeric to character conversion**: release 4.x float-to-character conversion allowed some truncation without warning. In release 10.x and 11.x, all conversions to character succeed only if no decimal digits are lost. Previously, floating point to character conversions allowed some truncation without warning.
- **Conversion to money**: All conversions to *money* datatypes round to four places.
- **Explicit conversion from numeric to numeric**: When an explicit conversion of one numeric value to another results in loss of scale, the results are truncated without warning. For example, explicitly converting a *float* to an *integer* causes SQL Server to truncate all values to the right of the decimal point.
- **Integer to character conversion**: Conversions from integer to character return an error if an overflow occurs. They formerly returned a buffer of "*".

set arithabort/arithignore Behavior

The **set arithabort** and **set arithignore** commands have changed behavior in some cases. If you are using these commands, you may want to test and understand the new behavior.

For more about these commands, see *What's New In SQL Server 11?*

System Stored Procedures Output

Many of the existing system stored procedures provide new and improved output. If your current applications use system stored procedures, check to see what they report under SQL Server 11.x.

showplan Output

set showplan prints full command names, instead of abbreviations.

The output for **set showplan on** includes several changes. Any applications that rely on the output generated by this command may need to be changed.

For details about **showplan** messages, see the *Performance and Tuning Guide*.

Improved Error Messages

Many error messages are added, and some existing messages are changed to improve their readability. If you rely on the text of any error messages within your applications, check to be sure they have not changed. You can use the following statement to see the text changes on SQL Server 11.x:

```
select * from sysmessages where error = x
```

Database Administration Changes

This section tells you how to rename databases manually, if needed, and identifies a load database script change.

➤ ***Note***

For RS/6000 AIX sites, also see DBA information in Appendix F, "Create Database Devices on AIX with Additional Space."

Databases Online / Offline

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

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

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

sybsystemprocs Database

The *sybsystemprocs* database contains all system procedures. Actions to take include:

- You may need to increase the **number of open databases** configuration parameter. For details, see the *System Administration Guide*.
- Put *sybsystemprocs* under your regular backup plan. For more information about backups as of release 10.x and 11.x, see "Backup Server" on page 3-40.

Moving Your Stored Procedures

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

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

Procedures that affect *master* are not run from within a transaction. To prevent problems:

- Include a check, such as:

```
if @@trancount > 0
begin
   print "Can't run this procedure from within a
transaction"
  return 1
end
```

- Do not include any changes to *master* database tables within a transaction. Doing so can cause recovery problems on the *master* database.

System Administration Changes

This section covers release 10.x and 11.x changes that may impact your system administration scripts.

New Login and Password Protocols

Login / password changes include the following:

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

➤ ***Note***

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

New *create table* Permission

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

Run File Renamed

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

Changes you may need to make are as follows:

- If you use the **startserver -fRUNSERVER** command to start your server, you must change it to **startserver -fRUN_SYBASE.**
- If you start the server automatically when you boot your machine, change your system startup file if necessary.

New Approach for Storing Configuration Values

Prior to release 11.x, SQL Server stored configuration values in the first page of the master device, known as the “configuration block”. Most of these values have been moved to a flat file known as the configuration file.

Configuration File Administration

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

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

reconfigure Command Ignored

The **reconfigure** command is no longer required after running **sp_configure**. Any of your scripts that include **reconfigure** will continue to work; the **reconfigure** command is ignored. You may want to consider removing **reconfigure** commands from your scripts now, however, to prevent problems in future releases.

Trace Flags

The following table shows the trace flags that are now configuration parameters:

Old Name	New Name
T1204	print deadlock information
T1603	allow sql server async i/o
T1610	tcp no delay
T1611	lock shared memory

Check whether you are currently using trace flags that have been converted to the configuration file and, if so, reset them after you upgrade.

If you currently set any of these trace flags in your run file, set them one of the following ways:

- Use **sp_configure,** or
- Edit the configuration file.

For example, if you use a trace flag to print deadlock information to your error log (**-T1204** in your runserver file on UNIX), do both of the following:

- Remove it from your runserver file.
- Set deadlock information printing with **sp_configure** or by editing your configuration file.

For details on configuration parameters, see the *System Administration Guide.*

buildmaster Flags

The **buildmaster** executable no longer supports the **-y** and **-r** flags. Use **sp_configure** or the configuration file in place of these flags. If you have scripts that use these flags, rewrite them using **sp_configure**, or save and edit copies of the configuration file.

For details about the configuration file and **sp_configure** options, see the *System Administration Guide.*

Configuration Parameter Changes

SQL Server 11.x includes new and changed configuration parameters. This section addresses only a few. For more information:

- *What's New in Sybase SQL Server Release 11.0?* lists the new and changed parameters.
- The *System Administration Guide* details how to use the configuration parameters.

deadlock checking period Parameter

Prior to release 11.0, SQL Server initiated deadlock checks as soon as a task had to wait for a lock. To keep this behavior, set the **deadlock checking period** to 0 (zero) in the configuration file.

SQL Server 11.x performs deadlock checking after a minimum period of time for any process waiting for a lock to be released. By default, this minimum period is 500 milliseconds, but you can change this default with **deadlock checking period**.

Delaying deadlocking saves overhead. If you expect your applications to deadlock infrequently, you can reduce overhead cost by delaying deadlock checking. However, increasing **deadlock checking period** causes longer delays before deadlocks are detected.

For details about deadlock checking, see the *Performance and Tuning Guide.*

page utilization percent Parameter

The configuration parameter **page utilization percent** saves time by allocating new extents rather than searching the OAM page chain. This parameter makes page allocation faster, but it may waste space.

Prior to release 11.x, SQL Server searched the OAM page chain for unused pages before allocating a new extent. To keep this behavior, use **sp_configure** to set **page utilization percent** to 100.

sp_configure Parameter Name Changes

Some of the configuration parameters available through **sp_configure** have new names. For example:

- Memory is now called **total memory**. Running **sp_configure memory, *value*** results in this error: "Configuration option is not unique."
- The **allow updates** parameter name is changed to **allow updates to system tables**. To ensure that your scripts work, change any that use the syntax **sp_configure allow** to one of the following:

  ```
  sp_configure "allow updates", option
  ```

  or

  ```
  sp_configure "allow updates to system tables",
  option
  ```

If you have scripts that use **sp_configure** to set or report on configuration parameters, change them if the parameter names have changed. Be sure to test all scripts that use the changed names.

Increased Memory and Cache Needs

SQL Server 11.x uses more memory for the Sybase kernel and for internal structures including the new user log cache. You may need to add more total memory to your server to maintain the same performance as your previous release.

Also, compiled objects grew about 75 percent in release 10.x and about 25 percent in release 11.x. You may need to enlarge your procedure cache to maintain the same performance. Use the following rule of thumb to estimate the memory increase needed:

For	Memory Increase
Kernel and internal structures	5MB (could be more for multi-engine)
User connections	30K
Stored procedures	100 percent

➤ ***Note***

Named data caches allow you to use memory more effectively, which requires memory in addition to these rule-of-thumb estimates.

For more information, see the *Performance and Tuning Guide*.

Decide Last Chance Threshold Behavior

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

You need to decide how to use the last chance threshold before you migrate your production system. The default behavior suspends action when the last chance threshold on the transaction log is reached, and all users hang until some action is taken, instead of getting the 1105 error.

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

For an understanding of managing free space with thresholds, see the *System Administration Guide*.

Sample Procedure

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

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

Open Transactions

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

➤ ***Note***

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

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

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

Troubleshooting

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

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

Tip for Bulk Copy Users

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

Backup Server

This section covers considerations for employing the new Backup Server in your backup strategy.

Handling of Dumps and Loads

As of release 10.x and 11.x, dumps and loads are handled by a separate process called the Backup Server. When upgrading your database, you must install a Backup Server through the install procedure (**sybinit** or **Server Config**), or you will be unable to do dumps and loads on SQL Server 11.x.

➤ ***Note***

Backup Server runs in addition to SQL Server, which requires extra machine resources.

During a dump or load on UNIX platforms, Backup Server starts additional small processes called **sybmultbuf** to communicate with database and dump devices.

Recommended Testing

Run some tests with dump and load to be sure your scripts work and to monitor the additional machine resources needed to do a dump or load on your system.

Recovery procedures on system databases have changed with the Backup Server and the *sybsystemprocs* database. Consult the *System Administration Guide,* and test your procedures for recovery.

Automatic Remote Connections

Remote connections are automatically enabled when SQL Server is upgraded or installed so that SQL Server can communicate with Backup Server. However, the **allow remote access** configuration parameter is now dynamic, so you no longer have to reboot SQL Server to change its behavior.

Start and Stop Required

You must start and stop Backup Server, as well as SQL Server. The install or upgrade program sets up the appropriate scripts to start Backup Server.

An option in the **shutdown** command stops Backup Server:

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

Initiate your own procedures for starting and stopping the Backup Server process as appropriate.

➤ ***Note***

If you use threshold procedures to **dump transactions** to a device, remember to keep Backup Server running at all times.

Interfaces File Entries

You must maintain entries in the interfaces file for the Backup Servers in use. The install or upgrade program makes the necessary entry for your local machine, but you may need to determine if those entries need to be distributed to other copies of the interfaces file.

Changes to Dumps

This section covers the changes to dump handling that may affect your dump scripts.

No Null Device

The device */dev/null* on UNIX or *NL:* on VMS will no longer be available. **sybinit** removes the default entries for these devices from the *sysdevices* table.

Make sure that any dump scripts do not use one of these default devices or any other dump device that points to */dev/null* or *NL:*. If you do not change these scripts, dumps fail with the following error:

```
Backup Server: 4.56.2.1: Device validation error:
couldn't obtain tape drive characteristics for
device /dev/null, error: Invalid argument
```

Dump Script Behavior

The single feature that can most affect your current use of tapes and dump commands is Backup Server's ability to make multiple dumps to a single tape. The new dump is placed after the last existing file on the current tape volume.

Be aware of the following changes when backing up databases immediately after upgrading:

- If you use new tapes, or tapes without ANSI labels, your pre-10.0 dump scripts overwrite the entire tape.
- If you use single-file media (for example, quarter-inch cartridge) with ANSI labels, and the expiration dates on the tapes have expired, pre-10.0 dump scripts will overwrite the tapes.
- If the expiration date on a single-file tape has not been reached, you will be asked to confirm the overwrite; a positive response will overwrite the existing tape; a negative response initiates a request for a volume change, and tests are repeated on the new volume.
- If you use multi-file tape media, and do not change your dump scripts, the dump will be appended to the existing files on the tape.
- If you want to overwrite existing tapes that have ANSI labels, you must append the **with init** clause to existing **dump** commands:

  ```
  dump database mydb
      to datadump1
      with init
  ```

  You can also use operating system commands to erase or truncate the tape.

No Dumps from Within User-Defined Transactions

dump database and **dump transaction** are no longer allowed within a user-defined transaction.

No dump transaction to Device After Non-Logged Text Writes

dump transaction to a device is no longer allowed after a non-logged text operation.

If you use **dump transaction** after non-logged text writes on the same database, you must change the text writes to be logged. Otherwise, you will be unable to use **dump transaction** as part of your backup scheme.

For details on changing your text writes, see the following as needed:

- *SQL Server Reference Manual* (**writetext** command)
- *DB-Library Reference Manual* (**dbwritetext()**, and so on)
- *Open Client Client-Library Reference Manual* (**ct_send_data()**, and so on)

Loading from Multiple Tape Devices

A second connection to SQL Server is now required when loading a database to a tape device that spans multiple tapes. This connection allows you to issue the **sp_volchanged** stored procedure, which notifies Backup Server that the tape operator has finished handling a volume, such as changing a tape.

If you execute the command **load database master** from a tape device that requires a volume change, you need a second running SQL Server to issue the **sp_volchanged** stored procedure. Because SQL Server must be in single user mode to load *master*, you cannot log in a second time to send the volume change request. For this reason, Sybase Technical Support strongly recommends that you do one of the following:

- Ensure your tape device has enough space to hold the full dump of *master* before you dump the *master* database, or
- Use a disk device for your *master* backups.

Preview Performance

This section covers tasks that help you preview the performance of the new release. First, it may be helpful to review the information on testing in Chapter 1, “Planning Migration.”

Execute Tests

Execute both your application test suite to validate results and performance scripts to measure performance goals.

If you are using only one machine to migrate from one system to another and you are doing the migration and validation on a weekend or off hours:

- Arrange to have weekend or off hours support from Sybase, as well as the operating system and hardware vendors.
- Consider mirroring SQL Server in addition to having a pre-migration backup. For more information about mirroring disk devices, see the Sy*stem Administration Guide*.

Compare Release 11.x Tests to Release 4.x Baseline

Once you have established a release 4.x baseline as described in Chapter 1, "Planning Migration," create a release 11.x test system. Run test scripts, and compare the outputs between the two releases.

The steps include:

1. Upgrade the test server to release 11.x, as described in this chapter. This important step allows you to catch and resolve problems before doing the production system upgrade.
2. Perform database consistency checking on all release 11.x test databases, including *master* and *sybsystemprocs*. Run **dbcc** with options **checkdb**, **checkalloc**, and **checkcatalog**.
3. Back up the databases.
4. Run single-user tests on the release 11.x test system to validate optimizer decisions as follows:
   - Capture **showplan**, **statistics io**, **statistics time**, and **dbcc traceon(302,310)** output.
   - Compare the output to that of the release 4.x test system. Look for the same or better optimizer decisions, and the same logical and physical I/O counts.
   - Fix all query plan problems now. Otherwise, they may be difficult to distinguish from other problems that surface later. Unresolved query problems also can invalidate multi-user tests, requiring extra problem-solving and testing time.
5. Iteratively run multi-user tests for each component as follows:
   - Capture the transaction throughput and response metrics.

- Compare the throughput and metrics to those of the release 4.x test and productions systems.
- Resolve obvious problems.
- Note opportunities for tuning. For information about the various layers for tuning, see the *Performance and Tuning Guide*.
- To verify that there is no data corruption, periodically run **dbcc** with options **checkdb**, **checkalloc**, and **checkcatalog**. This is not necessary after every test.
- Restore the databases to a known state, reloading them from backup.
- Re-run the tests until you are satisfied.

6. Back up the release 11.x test system databases.

Run Followup Tests

Execute the following tests against the release 11.x test system:

- Integration tests, to make sure that all the system components work together
- End-user acceptance tests to verify that user standards are satisfied
- Upgrade tests and fallback plan tests to make sure no steps are missing and contingencies are covered

For more information about these tests, see Chapter 1, "Planning Migration."

Performance Tuning Tips

The *Performance and Tuning Guide* provides in-depth system administration, database design, and application programming information.

To immediately enhance SQL Server 11.x functions, such as subqueries, **bcp** function, dumps, **dbcc** commands, **create index** statements, and **select** statements with large result sets, perform these tuning actions:

- **Optimizer strategy**—Determine whether the optimizer is choosing the new MRU strategy correctly. See the *Performance and Tuning Guide* for information about:

  - The **forceplan** option and specifying the index for a query
  - The **sp_cachestrategy** procedure for specifying cache strategy

- **Index choice**—Ensure that you have the right set of indexes on your tables, so that queries access data with a minimum number of page reads. For details on indexing, see the *Performance and Tuning Guide*.
- **Memory/cache allocation**—Increase SQL Server 11.x memory to break even with that of your current system. Do not set memory too low. See "Increased Memory and Cache Needs" on page 3-37.
- **Performance monitoring**—Use **sp_sysmon**, the stored procedure that produces SQL Server performance data that you can use for tuning. For details on monitoring performance with **sp_sysmon**, see the *Performance and Tuning Guide*.

If performance drops after upgrading from release 10.x to 11.x, research these options:

- Put subquery changes into effect. See "Drop and Re-create Compiled Objects Containing Subqueries" on page 3-21.
- Prevent the housekeeper task from increasing database writes. Set **housekeeper free write percent** to 0 (zero). See the *System Administration Guide*.

Practice!

Explore changes from release 4.x, such as:

Changed or New	For More Information
Backup Server	*System Administration Guide*
Thresholds	*System Administration Guide*
ANSI compliance	*What's New in SQL Server Release 11.0?* (chapter 2)
Cursors and IDENTITY columns	*SQL Server Reference Manual* *Transact-SQL User's Guide*
Security: • Use **sp_locklogin**. • Set a password with less than 6 characters. • Assign appropriate roles.	*Security Administration Guide*

Changed or New	For More Information
Named data caches	*Performance and Tuning Guide*
Large I/O	*Performance and Tuning Guide*
Query tuning options	*Performance and Tuning Guide* *Analyzing, Handling, and Resolving Optimizer Problems/Symptoms* (in the Technical Information Library on the Internet)
Configuration file and **sp_configure**	*System Administration Guide*
sp_sysmon to monitor counters and analyze performance	*Performance and Tuning Guide*
Reorganized documentation for:	See the following, respectively:
• System table information	• *Reference Supplement*
• Security information	• *Security Administration Guide*

For a more detailed list of changes, see *What's New in Sybase SQL Server 11.0?*

A Worksheets—Your Current Environment

Before migration, appraise your existing resources and profile the applications that use those resources.Such information helps you plan migration timing, based on an application's needs.

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

High-Level View of Production Environment

Identify all the system components in the current SQL Server production environment. You can use a diagram, narrative, table, or combination of all three. This high-level view serves as common ground for everyone in identifying the issues and priorities, as well as the needed resources.

Ensure that your production environment survey includes:

- Clients and servers. Identify each server by:
  - Machine name and IP address
  - Sybase name from the interfaces file and listener ports
  - Application names and the number of users connected to the server
- Required file, print, and application servers
- Gateways, routers, brouters, bridges, and so on, as well as protocols

SQL Server Infrastructure Worksheet

The Infrastructure Worksheet includes these sections:

Record this information for both the **production** and **development** environments. You may want to create a separate set of infrastructure worksheets for each environment.

Host Configuration

Record applicable data on your SQL Server host configuration.

Hardware

Identify the hardware manufacturer.

Table A-1: Host hardware

SQL Server Machine
Make:
Model:
Customer ID with hardware vendor:
Technical support phone number:
Technical support hours:
Technical account manager: • Name • Phone number (s) • Pager number

Identify the CPU resources..

Table A-2: CPU resources

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

Physical Memory Usage

List all the major processes and memory requirements running on each server.

Table A-3: Runtime memory usage

Application	Runtime Memory Usage
Operating system	
Networking software	
SQL Server memory parameter	
Open Servers Include: • Backup Server • Replication Server • Log Transfer Manager • Monitor Server • Gateways (list)	
Other applications	
Total memory required	
Total memory installed	

Disk I/O Configuration

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

Table A-4: Disk I/O

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

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

Table A-5: Disk and firmware use

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

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

Table A-6: Disk and partitions

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

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

Table A-7: Logical volumes

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

Network Configuration

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

Table A-8: Network layout

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

Tape Configuration

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

Table A-9: Tape layout

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

Operating System Configuration

Detail the operating system.

Table A-10: OS details

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

SQL Server Configuration

Document general information about the SQL Server configuration.

Table A-11: SQL Server configuration

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

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

Table A-12: dbcc output

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

Database Devices

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

Table A-13: Database devices

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

Databases and Segments

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

Table A-14: Databases and segments

Database Name	Database Options Set	Fragment (page range)	Size (in MB)	Segment Names	Device Name
master	none	0-1535	3	default, system, log	*master*
master		1536-2559	2	default, system, log	*master*
master		2560-3071	1	default, system, log	*master*
model	none	0-1023	2	default, system, log	*master*
tempdb	**select into bulkcopy**	0-1023	2	default, system, log	*master*

Dump Devices

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

Table A-15: Dump devices

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

SQL Server Operational Worksheet

The information you provide in the SQL Server Operational Worksheet represents business requirements.

This worksheet includes the following sections:

Operational Business Requirements

General operational information can help with documentation and success criteria.

Table A-16: Operational requirements

Database Name	Operational Hours	Maximum Downtime	Comments

Backup and Restore Procedures

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

Table A-17: Backup/restore procedures

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

Database Dump Details

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

Table A-18: Backup/restore details

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

Maintenance Procedure Details

Detailed maintenance information is helpful for defining success criteria.

Table A-19: Maintenance

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

Data Architecture Worksheet

The Data Architecture Worksheet includes these sections:

Client Application Components

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

Table A-20: Client applications

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

Production Performance Metrics

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

- CPU

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

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

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

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

Transaction Profile

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

Table A-21: Transaction profile

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

➤ ***Note***

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

B Sample Migration Task Lists

This appendix shows sample task lists, including a general list and lists specific to each migration approach. The sample task lists are:

➤ ***Note***

First use your task list to create a **test** environment, make appropriate application changes, and run test suites. Do not move to a **production** environment until satisfied with the test environment.

General Task List Example

The following sample lists typical tasks for migrating databases and client applications for rebuilding SQL Server.

Table B-1: Sample rebuild tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Prepare preliminary information.						
Analyze your environment. See Appendix A, "Worksheets—Your Current Environment."						
Determine migration feasibility.						
Define test / acceptance criteria.						
Prepare *database_name* migration.						
Complete the worksheet in the SQL Server installation guide.						
Record SQL Server 10.x or 4.x configuration values.						

Table B-1: Sample rebuild tasks (continued)

Task	Date	Begin Time	End Time	Duration	Owner	Status
Record SQL Server 10.x or 4.x dump devices.						
Record data segmentation / disk allocation per database.						
Retrieve database creation scripts per database. **Note:** Remember to save the key databases in *master*: *syslogins*, *sysdatabases*, *sysdevices*, and *sysusages*.						
Assess resource requirements per database.						
Develop hardware and SQL Server configuration plan and scripts.						
Configure the new hardware / devices and document them.						
Download and install pre-11.x SQL Server, SQL Server 11.x, and Backup Server.						
bcp out *syslogins*.						
Modify system administration scripts.						
Prepare for application migration.						
Verify compatibility of non-Sybase applications with SQL Server 11.x.						
Record the database options.						
Create the test plan, user acceptance scripts, and validation test scripts.						
Create the performance test script.						

Table B-1: Sample rebuild tasks (continued)

Task	Date	Begin Time	End Time	Duration	Owner	Status
Run the validation and performance test scripts for benchmarks.						
Freeze the development of application code.						
Create your fallback method, such as **bcp** scripts.						
Copy stored procedures, triggers, views, and application code to test the new directory.						
Evaluate the subqueries in stored procedures, triggers, and views.						
Document the needed changes in these subqueries.						
Evaluate the subqueries in client applications.						
Document and make the required changes to application subqueries.						
Search for keywords in client applications for Release 4.x or 10.x and Release 11.x. See the topic on using **sp_checkreswords** in (chapter 2 or 3, as appropriate.						
Search for **set arithabort** and **set arithignore** in stored procedures, views, and triggers.						
Search for **set arithabort** and **set arithignore** in client applications.						
Verify new datatype conversion and computation in stored procedures, views, and triggers.						
Search for queries that specify correlation names (aliases).						

Table B-1: Sample rebuild tasks (continued)

Task	Date	Begin Time	End Time	Duration	Owner	Status
Search client applications for the use of **select into** with NULL column headings.						
Determine whether applications rely on Sybase error messages.						
Determine and apply changes to applications.						
Migrate databases to your test system.						
Create the databases.						
bcp in *syslogins*.						
Run **dbcc** commands on data to be migrated.						
Dump the original 4.x or 10.x databases.						
Load the databases into a new 4.x or 10.x server.						
Run **sybinit** to upgrade the server to Release 11.x.						
Identify and apply changes to Release 10.x or 4.x database objects for reserved words. See the topic on checking for reserved words in chapter 2 or 3, as appropriate.						
Modify environment variables for SQL Server 11.x.						
Change stored procedures, triggers, views, and subqueries in Release 11.x databases.						
Run the validation test scripts.						
Modify application code, if necessary.						
Reload test system databases, if necessary.						
Re-run the validation test scripts, as needed.						

Table B-1: Sample rebuild tasks (continued)

Task	Date	Begin Time	End Time	Duration	Owner	Status
Perform user acceptance test.						
Do post migration tasks.						
Run **dbcc** commands.						
Back up the new databases.						
Perform nightly **bcp** jobs for fallback processing.						
Install the new version of application code on all front ends.						
Change applications to access SQL Server 11.x.						
Install the new system administration scripts.						
Complete migration documentation.						
Address performance and tuning needs.						
Measure SQL Server 11.x resource usage.						
Analyze performance test results.						
Implement minimum immediate tuning.						
Re-run performance test scripts, if necessary.						
Perform additional performance and tuning. See the *SQL Server Performance and Tuning Guide*.						

Parallel Migration Task List Example

This sample task list covers a parallel migration with replication approach to upgrade SQL Server from Release 10.x to 11.x.

The scenario for this task list is as follows:

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

Prepare Preliminary Information

Table B-2: Sample preliminary tasks for parallel migration

Task	Date	Begin Time	End Time	Duration	Owner	Status
Analyze your environment. See Appendix A, "Worksheets—Your Current Environment."						
Determine migration feasibility.						
Make go/no go call.						

Define Test/Acceptance Criteria—Regression Test Suites

Table B-3: Sample test preparation tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Back-end regression test suite—production loads						
Identify back-end queries.						
Encapsulate back-end queries.						
Create back-end test suite (**showplan**, **stat io**, and **stat time** wrappers).						
Front-end simulation regression test suite						
Identify target user functions.						
Capture and map SQL for target user functions.						
Encapsulate SQL for user functions.						
Create front-end simulated test suite (**showplan**, **stat io**, and **stat time** wrappers)						
Front-end Phase 1 and 2 regression test suite						
Identify front-end test scenarios.						
Understand front-end application and acceptance / test procedures.						
Document functional test approach.						
Compose front-end test mix matrix.						

Set Up Target Production Environment

Table B-4: Sample setup tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Identify physical drive configuration for Server A (production) and Server B (shadow).						
Configure physical drives.						
Perform a dump of the production environment.						
Install SQL Server 10.x on Server B.						
Configure SQL Server 10.x on Server B, to duplicate Server A.						
Update the interfaces.						
Create the databases.						
Load the Server A dump on Server B.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**).						
Synchronize user IDs.						
Run **checkpoint**.						

Set Up Replication Server

Table B-5: Sample replication setup tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Install Replication Server on Server B.						
Configure Replication Server on Server B.						
Install Replication Server on Server A.						

Table B-5: Sample replication setup tasks (continued)

Task	Date	Begin Time	End Time	Duration	Owner	Status
Configure Replication Server on Server A (secondary).						
Verify replication functionality between the two servers: Test replication on target objects. Verify and checkpoint replication-ready environment.						
Recycle SQL Server.						

Run Regression Test Suites

Table B-6: Sample regression testing

Task	Date	Begin Time	End Time	Duration	Owner	Status
Back-end regression test suite—production loads						
Update back-end scripts.						
Iteratively perform back-end regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock, statistics io**...).						
Verify and document SQL Server 10.x regression test.						
Front-end simulation regression test suite						
Iteratively do front-end simulation regression tests.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document SQL Server 10.x regression test.						
Front-end Phase 1 and 2 regression test suite						
Perform Phase 1 and 2 regression testing for local team.						
Perform Phase 1 and 2 regression testing for users.						
Monitor and capture dynamics (**sp_who**, **sp_lock**...).						
Make go/no go call for the shadow SQL Server 10.x.						
Verify and document SQL Server 10.x regression test.						
Verify performance and functions on Server B's SQL Server 10.x.						

Upgrade SQL Server on Server B (Shadow)

Table B-7: Sample upgrade tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Alter *sybsystemprocs*.						
Make the pre-upgrade verification.						
Upgrade Server B to SQL Server 11.x.						
Load SQL Server 11.x environment from media dumps.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**).						
Set baseline SQL Server 11.x configuration parameter values.						
Perform Release 10.x dump from Server A.						
Load SQL Server 11.x with the Release 10.x dump.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on SQL Server 11.x databases. Be sure to verify the **dbcc** log and errorlog.						
Recycle SQL Server.						

Run Regression Test Suites on Server B

Table B-8: Sample post-upgrade regression testing tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Back-end regression test suite—production loads						
Perform the back-end regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**, **statistics io**...).						
Verify and document this SQL Server 10.x regression test.						
Front-end simulation regression test suite						
Perform front-end simulation regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document this SQL Server 10.x regression test.						
Load SQL Server 11.x from media dump.						
Front-end Phase 1 and 2 regression test suite						
Perform Phase 1 and 2 regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document this SQL Server 10.x regression test.						
Other Testing						
Verify 11.x performance and functions.						

Run User Acceptance Tests on SQL Server 11.x (Server B)

Table B-9: Sample user acceptance testing tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Production testers perform user regression testing of shadow Release 11.x.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document this SQL Server 11.x regression test.						
Determine if Release 11.x is go / no go.						

Shift Production Users to SQL Server 11.x (Server B)

Table B-10: Sample tasks to move to production

Task	Date	Begin Time	End Time	Duration	Owner	Status
Ensure that there is no production activity.						
Perform a Release 10.x dump from Server A (production).						
Load the dump onto Server B (shadow).						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on the Release 10.x databases just loaded to Server B. Be sure to verify the **dbcc** log and errorlog						
Switch the IP address, and rename the machines and servers.						
Run user testing and verification.						

Perform Final Steps

Table B-11: Sample final tasks

Task	Date	Begin Time	End Time	Duration	Owner	Status
Enable replication (Server B to Server A).						
Start production users on Release 11.x (Server B).						

Cutover Migration Task List Example

This sample task list covers a cutover migration approach to upgrade SQL Server from Release 10.x to 11.x. The scenario for this task list is as follows:

- 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 1 hour and cannot afford more than 8 hours of data loss.
- The environment includes development, test, and production systems.

Prepare Preliminary Information

Table B-12: Sample preliminary tasks for cutover migration

Task	Date	Begin Time	End time	Duration	Owner	Status
Analyze your environment. See Appendix A, "Worksheets—Your Current Environment."						
Determine migration feasibility.						
Make go / no go call.						

Set Up SQL Server 11.x Environment on Development System

Table B-13: Sample setup tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Configure the physical drives and local array. Duplicate the environment of the Release 10.x development system.						
Perform a dump of the Release 10.x development system.						
Install SQL Server 11.x.						
Configure SQL Server 11.x on the development system, duplicating the Release 10.x development environment.						
Update the interfaces.						
Create the databases on SQL Server 11.x.						
Load the dump of the Release 10.x development system.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on SQL Server 11.x databases. Be sure to verify the **dbcc** log and errorlog.						
Synchronize user IDs between the Release 10.x and 11.x development systems.						
Run **checkpoint** on the Release 11.x environment.						
Shift development to SQL Server 11.x. Developers begin verification and new feature development.						

Define Test/Acceptance Criteria—Regression Test Suites

Table B-14: Sample test definition tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Front-end simulation regression test suite						
Identify target user functions.						
Capture and map SQL for target user functions.						
Encapsulate SQL for user functions.						
Create front-end simulated test suite (**showplan**, **stat io**, and **stat time** wrappers)						
Front-end regression test suite						
Identify front-end test scenarios						
Understand front-end application and acceptance/test procedures.						
Document functional test approach.						
Compose front-end test mix matrix.						

Define Release 11.x to 10.x Fallback Procedures on Test System

Table B-15: Sample tasks to define fallback procedures

Task	Date	Begin Time	End time	Duration	Owner	Status
Create the fallback scripts to re-create the Release 10.x environment.						
Document fallback procedures.						

"Baseline" Release 10.x Environment on Test System

Table B-16: Sample task to establish baseline

Task	Date	Begin Time	End time	Duration	Owner	Status
Recycle SQL Server 10.x.						

Run Regression Test Suites on Release 10.x Test System

Table B-17: Sample regression testing tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Front-end simulation regression test suite						
Iteratively perform front-end simulation regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document this SQL Server 10.x regression test.						
Front-end regression test suite						
Perform regression testing for local team.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Make the go / no go call for the Release 10.x test system.						
Verify and document this SQL Server 10.x regression test.						
Verify performance and functions on the test SQL Server 10.x.						

Upgrade Release 10.x Test System to Release 11.x

Table B-18: Sample upgrade tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Perform a dump of the Release 10.x databases.						
Before the upgrade, run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**).						
Alter the *sybsystemprocs* database.						
Perform a pre-upgrade verification.						
Upgrade the test system to Release 11.x.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on the Release 11.x databases. Be sure to verify the **dbcc** log and errorlog.						
"Baseline" the Release 11.x configuration.						

Run Regression Test Suites on Release 11.x Test System

Table B-19: Sample regression test run tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Recycle SQL Server.						
Back-end regression test suite—production loads						
Iteratively perform the back-end regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock, statistics io**...).						
Verify and document SQL Server 11.x regression test.						
Front-end simulation regression test suite						
Perform front-end simulation regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Make the go / no go call for the Release 11.x system.						
Verify and document this SQL Server 10.x regression test.						
Front-end regression test suite						
Perform user regression testing.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document SQL Server 11.x regression test.						
Other Testing						
Verify 11.x performance and functions on the test system.						

Run User/Acceptance Tests on the Release 11.x Test System

Table B-20: Sample user acceptance testing tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Testers perform user regression testing of Release 11.x.						
Monitor and capture system dynamics (**sp_who**, **sp_lock**...).						
Verify and document this Release 11.x regression test.						
Determine if Release 11.x is go/no go for the production system.						

Execute Release 11.x to 10.x Fallback Procedures on Test System

Table B-21: Sample fallback execution tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Shut down the Release 11.x test system.						
Re-create the Release 10.x test system.						
Re-create the Release 10.x environment, using your re-creation scripts and procedures.						
Load the Release 10.x production system dumps.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on the Release 10.x databases. Be sure to verify the **dbcc** log and errorlog.						
Run **checkpoint** on the Release 10.x test system.						

Upgrade Production SQL Server from Release 10.x to 11.x

Table B-22: Sample production upgrade tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Perform a dump of Release 10.x databases on the production system.						
Before the upgrade, run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**).						
Alter the *sybsystemprocs* database.						
Perform a pre-upgrade verification.						
Upgrade the production system to Release 11.x.						
Run **dbcc** commands (**checktable**, **checkalloc**, **checkcatalog**) on the Release 11.x databases. Be sure to verify the **dbcc** log and errorlog.						
"Baseline" the Release 11.x configuration on the production system.						
Perform user testing and verification.						

Perform Final Steps

Table B-23: Sample final tasks

Task	Date	Begin Time	End time	Duration	Owner	Status
Start production users on the Release 11.x production system.						
Upgrade the test system to Release 11.x.						

Staged Cutover Migration Task List Example

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

- The large company has a mature client / server environment with multiple applications residing on a single SQL Server.
- In the event of failure, the company depends on nightly backups and transactions dumps. The system cannot be down more than 1 hour and cannot afford more than 2 hours of data loss.
- The environment includes development, test, and production platforms. It requires enough space for duplicate SQL 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 as follows:

1. Configure SQL Server 11.x on the development system, duplicating the SQL Server 10.x development configuration.
2. Migrate the development objects to the Release 11.x development system.
3. Construct regression test suites.
4. Construct **bcp** scripts to move object deltas.
5. “Baseline” the Release 10.x test environment.
6. Configure a duplicate SQL Server 11.x on the test platform.
7. Migrate the test / acceptance objects to the Release 11.x test system.
8. Conduct regression test suites on the test system.
9. Verify synchronization of objects between the Release 10.x and 11.x test systems.
10. Conduct user acceptance testing on the Release 11.x test system.
11. Configure a duplicate SQL Server 11.x on the production platform.
12. Migrate the production objects to the Release 11.x production system.

13. Move the production users to the Release 11.x production system.
14. Use your **bcp** scripts to resynchronize objects between the Release 11.x and 10.x production systems.

C Recovery from Upgrade Failure

If you have trouble completing your upgrade to System 11, this appendix helps you locate the task where the upgrade failed, diagnose the problem, and then complete the upgrade.

This appendix covers the following topics:

Summary of Upgrade Recovery Process

The general process to recover from upgrade failure is:

1. Exit the **sybinit** session.
2. Find out where the upgrade failure took place. What task was the upgrade process performing?
3. Determine what caused the task to fail and fix the problem.
4. Complete the upgrade with an appropriate procedure.

This document provides information on determining where the upgrade failed, guidelines for fixing the problem, and instructions on completing the upgrade.

Checking Log Files

The first step in determining where the upgrade failed is to look in the log files. The upgrade process creates two types of log files:

- **sybinit** log
- SQL Server error log

sybinit Log

To identify what task the upgrade was performing when it failed, check the **sybinit** log. The **sybinit** log (*$SYBASE/init/logs*) lists the procedures performed before the upgrade stopped.

The table under "Relating Messages to Stages of the Upgrade" shows the critical messages that reflect the stages of the upgrade.

SQL Server Error Log

Check the SQL Server error log. The error log shows error messages, stack traces, and other messages. By default, the error log is in *$SYBASE/install/errorlog*.

Cross-reference the error messages in the error log with the **sybinit** log.

See the *SQL Server Error Messages* manual for an explanation of common errors written to the SQL Server error log.

Relating Messages to Stages of the Upgrade

The following table shows the messages that mark the boundaries of the stages of the upgrade process. Find the latest one of these messages in the **sybinit** log file to determine at what stage the upgrade failed.

Table C-1: Messages and corresponding upgrade stages

If upgrade fails after this message	Upgrade stage was	For more information
`Unable to boot server 'servername'`	Starting SQL Server with the 11.x dataserver binary	"Starting SQL Server 11.x" on page C-4
`Upgrading SQL Server servername to release 11.x`	Upgrading system tables	"Upgrading System Tables" on page C-6
`Setting Upgrade Version to 110x`	Upgrading system tables complete	"Remapping Objects — 4.9.2 and Earlier Releases Only" on page C-9
`Marking upgrade as complete in all databases`	Upgrading online databases	"Upgrading Online Databases" on page C-8
`Remapping objects in database, master`	Remapping stored procedures and other objects	"Remapping Objects — 4.9.2 and Earlier Releases Only" on page C-9

Table C-1: Messages and corresponding upgrade stages (continued)

If upgrade fails after this message	Upgrade stage was	For more information
`Running task to boot the SQL Server`	Starting the 11.x SQL Server	Not applicable
`Running task to create the sybsystemprocs database`	Creating the *sybsystemprocs* database	"Installing sybsystemprocs Database" on page C-12
`Running task to install system stored procedures`	Running **installmaster**	Not applicable

Checking the Version

If you want to manually check the upgrade level of SQL Server, use the following query. It may be different than **select @@version**, which shows you which dataserver binary the server is using.

```
select value from sysconfigures
where config = 122
```

If the value is 110*x*, you cannot rerun the upgrade with **sybinit** or the upgrade script, because upgrade checks "upgrade version identifier" as first step.

In most cases where the upgrade failed and the upgrade version identifier indicates that SQL Server is at 11.x, the problem is either failure to remap objects or to create the *sybsystemprocs* database.

If this is the case, see "Remapping Objects — 4.9.2 and Earlier Releases Only" on page C-9 or "Installing sybsystemprocs Database" on page C-12 for instructions on manual remapping.

Fixing the Problem: General Guidelines

The following are general troubleshooting guidelines for upgrade problems:

- Verify that you followed all preparation procedures, including those from this guide's chapter 2 or 3, as appropriate to your environment, under "Prepare for Migration."
- As described in the section "Checking Log Files" on page C-1:
  - Use the log file messages to determine which **stage** the upgrade was in when it failed.
  - Use both the log and error file messages to determine what **task** was being done when the upgrade failed.
- Refer to the *SQL Server Troubleshooting Guide* for procedures to address specific problems and for information about error messages.

◆ ***WARNING!***

Many of the upgrade tasks update your system tables. Before attempting to fix a system table: 1) thoroughly understand the impact of any changes you may make, and 2) always bcp the data out from the table that you need to fix, if the table's format allows you to bcp out.

Troubleshooting the Upgrade Stages

The topics in this section describe the stages where the upgrade may fail, and the issues associated with each.

Starting SQL Server 11.x

The message "unable to boot server SYBASE 11.x" is displayed if the upgrade process cannot start a server with the 11.x **dataserver** binary.

Causes of Failure

Failure in this stage could be caused by one of the following problems:

- Insufficient operating system memory
- Port already in use

- Insufficient space for upgrade
- Memory problems
- Configuration
- Network problems, such as port can't start master install

Example: Messages When SQL Server Fails to Start

This example shows that SQL Server failed to start due to a memory problem. From the **sybinit** log file:

```
03/20/96 04:09:15 PM waiting for server
'GOLDEN_492' to boot...

03/20/96 04:10:08 PM SERVER ERROR: Failed to boot
server 'GOLDEN_492'.
```

From upgrade error log file:

```
00:96/03/20 16:09:17.45 kernel  os_create_region:
can't allocate -2147483648 bytes

00:96/03/20 16:09:17.46 kernel  kbcreate: couldn't
create kernel region.

00:96/03/20 16:09:17.46 kernel  kistartup: could
not create shared memory
```

Recommended Actions

If the upgrade fails at this stage, take these actions:

1. Check 11.x error log to find the errors that prevented startup.
2. Fix the problem that prevents SQL Server from starting.
3. Restart SQL Server using the old (pre-11.x) **dataserver** program.

◆ ***WARNING!***

Once sybinit has successfully started the 11.x SQL Server, do not restart SQL Server using the pre-11.x dataserver program or runserver file. If you do, SQL Server writes records to the transaction log; mixing records from different releases can cause data corruption.

4. Run **sybinit** again.

Upgrading System Tables

The message "Upgrading SQL Server *servername* to release 11.x" means **sybinit** has called the upgrade binary, which now makes updates to system tables. Failure during this stage is difficult and risky to fix.

Because failure at this stage can indicate data corruption, you may have to restore databases from backup. Evaluate the errors. If it makes sense, try to rerun the upgrade option of **sybinit**. If you cannot fix the problem and rerun **sybinit**, restore your databases from backup.

Restore from databases if you are not able to rerun the upgrade. Look at the symptoms. If the databases are corrupted, you have to go to the backup.

For details on restoring from backup, see the *System Administration Guide*.

Causes of Failure

Possible causes for upgrade failure during system table update are:

- SQL Server or the machine running the upgrade loses its network connection.
- The database in which you were working has run out of log or data space.
- You are not allowed to perform an update to a table.
- The database is corrupted.
- The database is not available (still in recovery.) It may be unavailable if the upgrade program begins to work before SQL Server finishes performing recovery after startup.

Example: Errors While Running the Upgrade Binary

In the following example, **sybinit** failed while upgrading SQL Server system tables. The upgrade failed while trying to create an index on *sysconfigures*, indicated by the last upgrade message and error message 1902 at the beginning of the output.

```
SQL Server message 1902, state 1, severity 16:
Cannot create more than one clustered index on table
‘sysconfigures’.  Drop the existing clustered index
‘sysconfigures’ before creating another
DB-LIBRARY error:  20018
General SQL Server error: Check messages from the SQL Server.
Starting upgrade to SQL Server 11.0.0.
Dropping system stored procedures from master.
Adding new messages to sysmessages.
Adding new columns to system tables in all existing databases.
Adding new sqlstates to sysmessages.
Changing name of system catalog columns in all existing
databases.
Executing following commands in all existing databases.
Set status2 field to 0.
Checking and updating status2 field.
Creating system catalog: sysattributes and index.
Creating system catalog: syspartitions and indexes.
Adding sysattributes manager class to sysattributes
Adding valid attribute classes attribute to sysattributes
Adding valid attributes  attribute to sysattributes
Adding upgrade items class to sysattributes
Adding add sysattributes attribute to sysattributes
Adding sysdatabases 10.0 cols attribute to sysattributes
Adding sysdatabases 10.0.1 maxlen attribute to sysattributes
Adding add sysreferences.frgndbname attribute to sysattributes
Adding add sysreferences.pmrydbname attribute to sysattributes
Adding drop sysreferences.csysref attribute to sysattributes
Adding drop sysreferences.ncsysref attribute to sysattributes
Adding drop sysreferences.nc2sysref attribute to sysattributes
Adding add sysreferences db names attribute to sysattributes
Adding add sysreferences.csysref attribute to sysattributes
Adding add sysreferences.ncsysref attribute to sysattributes
Adding add sysreferences.nc2sysref attribute to sysattributes
Adding revise syscomments.syscomments attribute to sysattributes
Adding sysindexes.maxrowsperpage attribute to sysattributes
Adding initialize maxrowsperpage attribute to sysattributes
Adding add syspartitions attribute to sysattributes
Adding allocation hints space  attribute to sysattributes
Adding checkpoint version 11.0 attribute to sysattributes
Adding transaction manager class to sysattributes
Adding transaction manager class to sysattributes
Adding Log I/O Size attribute to sysattributes
Adding buffer manager class to sysattributes
Adding cache binding attribute to sysattributes
Adding config manager class to sysattributes
Adding user config level attribute to sysattributes
```

```
Adding user config level attribute to sysattributes
Adding config parameter parent attribute to sysattributes
Adding lock strategy class to sysattributes
Adding Lock promotion attribute to sysattributes
Creating system catalog: syslogshold.
Update usertype to 6 for status column in sysconfigures.
Updating sysconfigures
Altering sysconfigures table
Dropping existing index from sysconfigures table
Creating new index on sysconfigures table
```

Recommended Actions

Continue the upgrade with one of the methods described below:

- Fix the problem if doing so does not involve making manual updates to system tables. For example, in the example above, you might try running **sp_fixindex**.
- Rerun the upgrade option of **sybinit.**
- If rerunning the upgrade option of **sybinit** fails or if your problem is data corruption, you must restore the earlier SQL Server from backups, then start **sybinit.**
- If you are unsure that you can fix a problem involving a system table, call Technical Support.

◆ ***WARNING!***

Making manual changes to system tables could leave them in an inconsistent state with respect to other system tables.

Restoring SQL Server from Backups

If you cannot fix the problem, or if rerunning the upgrade option of **sybinit** still fails, you must restore your databases from backups.

Refer to the installation or configuration guide for your platform for information about restoring from backup.

Upgrading Online Databases

The message “Marking upgrade as complete in all databases” indicates that the upgrade was in the process of online database

upgrade. During this stage, the SQL Server version is changed to the new 11.x version.

Causes of Failure

Possible causes for an upgrade failure during system table update include:

- SQL Server or the machine running the upgrade loses its network connection.
- The database in which you were working has run out of log or data space.
- The database is corrupted.

Example

In the following example, the upgrade failed during the online database upgrade stage:

```
Database ‘pubs2’: beginning upgrade step: creating index (table
sysreferences, index csysreferences) [ID 1007]
Database ‘pubs2’: beginning upgrade step: creating index (table
sysreferences, index ncsysreferences) [ID 1008]
Database ‘pubs2’: beginning upgrade step: creating index (table
sysreferences, index nc2sysreferences) [ID 1009]
Database ‘pubs2’: beginning upgrade step: noting the present
database upgrade level  [ID 1015]
```

Recommended Actions

A failure during this stage of the upgrade does not require you to run the upgrade again. Follow these steps:

1. Fix the problem that caused online database upgrade fail.
2. Restart SQL Server.

Remapping Objects — 4.9.2 and Earlier Releases Only

The next stage of the upgrade installation is remapping compiled objects. SQL Server upgrade includes query tree remapping, which restructures all stored procedures, triggers, rules, defaults and views.

At this stage, the upgrade binary has completed. Therefore, you cannot rerun the upgrade option of **sybinit**.

The remapping stage is not necessary if you are upgrading from release 10.x. Go to the next stage, "Installing sybsystemprocs Database" on page C-12.

You have to remap stored procedures and other objects, if you are upgrading from 4.9.2 or earlier, and the upgrade fails after the following message:

```
Remapping objects in database, master
```

Causes of Failure

Possible causes of failure during stored procedure remapping include:

- *sybsystemprocs* size
- Stack size

Refer to your installation or configuration guide for information on recovering from stack size problems.

Example

If the upgrade fails during the remapping process, you should see error messages and output that specifies the database objects. For example:

```
Remapping objects in database, pubs2.
    Remapping object, typedflt.
    Remapping object, datedflt.
    Remapping object, phonedflt.
    Remapping object, pub_idrule.
    Remapping object, title_idrule.
    Remapping object, totalsales_trig.
```

Recommended Actions

You can no longer rerun the upgrade option of **sybinit** after the message:

```
Setting upgrade version to 110x
```

To continue:

- Start **sybinit**.
- Select the Remap Stored Procedures option.

- If this option does not succeed, use the script in the section "Remapping Script" on page C-11.

Remapping Script

◆ ***WARNING!***

Remapping stored procedures can take a long time. Do not interrupt the remapping process.

```
/*
 * Use following SQL cmd scripts as
 *     isql -Usa -P < this_file > out_file
 *
 * The out_file should contains all the objects that
 * need to be remapped. You then can run
 *
 *          isql -Usa -P < out_file
 * to finish remapping task.
 */

set nocount on
go

/*
 * Change DBNAME to your database name.
 */
use DBNAME
go

/*
 * Change DBNAME to your database name.
 */
print 'use DBNAME'
print 'go'

declare prep_remp_csr cursor for
select convert(varchar(30), id)  from sysobjects
where type = 'V' or type = 'P' or type = 'R'
or type = 'D' or type = 'TR'
go
declare @pid varchar(30)
declare @cnt int
select @cnt = 0
open prep_remp_csr
```

```
fetch prep_remp_csr into @pid
while(@@sqlstatus = 0)
begin
/*
 * Change DBNAME to your database name.
 */
  print "dbcc remap ( %1!, DBNAME, 1)" , @pid
  print "go"
  if (@cnt < 10 )
  begin
   select @cnt = @cnt + 1
  end
  else
  begin
    select @cnt = 0
/*
 * Change DBNAME to your database name.
 */
    print "dump database DBNAME with truncate_only"
    print "go"
  end

  fetch prep_remp_csr into @pid
end

close prep_remp_csr
deallocate cursor prep_remp_csr
go
```

Installing *sybsystemprocs* Database

The message "Running task to create the sybsystemprocs database" means that **sybinit** is creating the *sybsystemprocs* database where the stored procedures will be installed.

If **sybinit** is unable to perform this step, you can perform it manually as follows:

1. Create a new device for the *sybsystemprocs* database of at least 19MB (21MB on Digital Unix).
2. Using **isql**, log into SQL Server and create the *sybsystemprocs* database with this command:

   ```
   1> create database sybsystemprocs
   2> on device_name = 19
   3> go
   ```

3. If *sybsystemprocs* already exists, alter it to 19MB. (Digital UNIX requires 21MB.)

alter database Problem on 10.x to 11.x Upgrades

If you are upgrading from 10.x and use **alter database** to change the size of *sybsystemprocs*, you may encounter an **alter database** bug. On some releases of 10.x, this bug causes **alter database** to extend only the data segment or only the log segment, rather than data and log, to the new database fragment. This can result in running out of log or data space.

For troubleshooting help, see the installation instructions.

Installing *master* Database, *model* Database, and System Stored Procedures

The message "Running task to install system stored procedures" means that **sybinit** is trying to install your system stored procedures.

If **sybinit** is unable to perform this step, you can perform it manually by running the following scripts:

```
isql -Usa -Psa_password -Ssvr_name
        -i$SYBASE/scripts/installmaster

isql -Usa -Psa_password -Ssvr_name
        -i$SYBASE/scripts/installmodel
```

D FAQs: Migrating SQL Server 4.x or 10.x on SunOS to 11.x on Solaris

The frequently asked questions (FAQs) in this appendix address migrating SQL Server 4.x or 10.x from SunOS to SQL Server 11.x on Solaris:

- Why did Sybase decide not to port SQL Server System 11 to SunOS?
- Is this an End-of-Life (EOL) Notice for Sybase products on SunOS?
- Can I still get support for my Sybase software on SunOS? How long?
- Is Sybase charging a fee to move SQL Server from SunOS to Solaris?
- How do I migrate SQL Server from SunOS to Solaris?
- Will I be able to run SunOS and Solaris at the same time and get supported on both? How long?
- Will Sybase continue to maintain SQL Server System 10 for SunOS, for example by issuing bug fixes? How long?
- Will Sybase provide consulting assistance during my migration?
- Are Sybase and Sun working together on SQL Server migration?

Why did Sybase decide not to port SQL Server System 11 to SunOS?

SQL Server 11 is a major release focused on quality, performance and scalability. As with any release of the SQL Server, we reviewed our operating environments to make sure that this release would be available on the platforms most requested by our customers.

Based on feedback from both customers and partners, we elected to offer the SQL Server 11 on Solaris only. We will continue to support Open Client and older versions of SQL Server on SunOS.

Is this an End-of-Life (EOL) Notice for Sybase products on SunOS?

No. Sybase will continue to do both of the following:

- Release new versions of Open Client for SunOS because our research shows that many customers will continue to use this platform for their existing SQL Servers and clients.

- Support SQL Server 4.9.x and 10.0.x on SunOS until we announce the End-of-Life of those releases.

Can I still get support for my Sybase software on SunOS? How long?

Yes. We will continue to support all Sybase products on SunOS, except SQL Server 11.x, until we announce the End-Of-Life (EOL) of specific releases. As far as existing releases are concerned, SunOS will receive the same support as other platforms.

Is Sybase charging a fee to move SQL Server from SunOS to Solaris?

No. A migration kit is available free to SunOS customers, which allows you to migrate your current SQL Server from SunOS (BSD) to SunSolaris 2.4 (SPARC). To place your order, call Sybase Customer Service at 1-800-8-SYBASE .

How do I migrate SQL Server from SunOS to Solaris?

The general process to migrate SQL Server from SunOS to Solaris is as follows:

- Prepare your SunOS system for migration to Solaris. See the following information:
  - SQL Server installation instructions
  - In this guide, chapter 2 or 3, as appropriate, for issues such as setting configuration parameters and performing backups
  - Your Sun documentation
- Perform the upgrade from SunOS 4.x to Sun Solaris 2.x. The primary source of information is chapter 10 in the *SQL Server Installation and Configuration Guide for Sun Solaris 2.x (SPARC)*.

  Release 4.x only: If migrating SQL Server 4.x to Sun Solaris, also see the information about using the **sybconfig** utility in the *SQL Server Installation Guide for Sun Solaris*, contained in the migration kit.
- Using the migration kit, which contains Solaris versions of SQL Server 4.9.2 and 10.0.2, install your current version of SQL Server on Solaris. For example, if you are now running Release 4.9.2 for SunOS, install Release 4.9.2 for Solaris.
- To upgrade your old SQL Server to Release 11.x, follow the upgrade instructions in the installation guide.

Will I be able to run SunOS and Solaris at the same time and get supported on both? How long?

Once you order the upgrade, you have 90 days to convert your system from SunOS to Solaris. During the 90 days Sybase supports SQL Server on both SunOS and Solaris at no additional charge.

Will Sybase continue to maintain SQL Server System 10 for SunOS, for example by issuing bug fixes? How long?

Sybase will continue to maintain System 10 for SunOS for the life of System 10.

Will Sybase provide consulting assistance during my migration?

Sybase is not offering a special consulting package for SunOS to Solaris migration. However, normal consulting services are available to you if you need help with the process. For information about services and costs, call 1-800-8-SYBASE.

Are Sybase and Sun working together on SQL Server migration?

Sybase and Sun are not offering a joint migration package. To find out if Sun is offering incentives for customers who wish to upgrade to Solaris, call your Sun vendor.

Will my third party applications run on SunOS?

Almost all third party applications run on both SunOS and Solaris. You may need to upgrade older third party applications to run on Solaris. Check with your application provider for Solaris availability.

E FAQs About SQL Server 11.x on Windows NT

This appendix answers frequently asked questions (FAQs) about installing and running SQL Server 11.x on Windows NT:

- Is it possible to install one Sybase product, for example SQL Server, without installing Net-Library, Open Client, or a language module?
- Is it possible to install both the 16- and 32-bit Open Client kits?
- Should I back up the registry after installing the SQL Server?
- Is the SyBooks documentation available on CD-ROM?
- How do I run SQL Server 10.x and SQL Server 11.x on the same Windows NT box?
- If I run both SQL Server 10.x and 11.x on the same Windows NT box, how do I supply different names for the servers?
- Is the SQL10 environment variable changing for release 11.x?
- What is the name of the default Sybase directory in SQL Server 11.x?
- Does including dashes in a Windows NT server name cause conflicts when running on a network that includes UNIX?
- Why is the default size for the system procedures database, sybsystemprocs, so much larger for SQL Server 11.x than for SQL Server 10.x?
- If I run both SQL Server 10.x and 11.x on the same Windows NT box, how do I supply different names for the servers?
- How do I give SQL Server a name other than the Windows NT machine name?
- Does including dashes in a Windows NT server name cause conflicts when running on a network that includes UNIX?
- What are the new minimum sizes for the master, system procedure, and auditing databases?
- Should I use NTFS or FAT file system for Sybase devices?
- Can I use virtual memory to increase SQL Server memory?
- Can I have more than one Backup Server or Monitor Server?
- Can I use domain login security in place of SQL Server login security?

- How can I schedule events to occur at a specific time?
- What do the entries in sql.ini stand for?
- When would I start SQL Server from the command line?
- How can I improve performance during functional suites?
- Are any Sybase features available only on Windows NT?
- Should I use Windows NT Workstation or Server for SQL Server 11.x?

Installing

Is it possible to install one Sybase product, for example SQL Server, without installing Net-Library, Open Client, or a language module?

No. If Open Client, Net-Library, and a language module are not already on your system, they will be installed when you install SQL Server. The Sybase products CD contains a single version of each of these products so that the correct version is installed for SQL Server and the other products that require them.

In earlier releases, Open Client, Net-Library, and a language module were bundled into such Sybase products as SQL Server and Monitor Server. Their installation was transparent to the user.

Because the Net-Library shipped with one Sybase product might be incompatible with that shipped with another, users sometimes installed inconsistent and redundant products.

Therefore, to enforce consistency among Sybase programs, the current installer loads the same version of all components for every program it installs.

Is it possible to install both the 16- and 32-bit Open Client kits?

Yes, if you copy the files into separate directories on your system.

See *Sybase SQL Server for Windows NT, Release 11.0.x*, under "Installing Sybase Products".

Is the SQL10 environment variable changing for release 11.x?

Yes. The SQL10 environment variable has been removed from SQL Server 11.x. Its functionality is incorporated into the SYBASE environment variable.

What is the name of the default Sybase directory in SQL Server 11.x?

The *sql10* directory was renamed *SYBASE* for SQL Server 11.x.

Should I back up the registry after installing the SQL Server?

Yes. Since the registry contains so much configuration information about Windows NT components and applications, it is always a good practice to back up the registry.

Is the SyBooks documentation available on CD-ROM?

Yes. The SQL Server documentation is shipped on the same CD-ROM as the software. You install it separately from the product installation.

Configuring and Administering

How do I run SQL Server 10.x and SQL Server 11.x on the same Windows NT box?

You install SQL Server 11.x in a separate directory from SQL Server 10.x. For detailed instructions, refer to *Configuring and Administering Sybase SQL Server for Windows NT.*

If I run both SQL Server 10.x and 11.x on the same Windows NT box, how do I supply different names for the servers?

To rename a SQL Server after it's been installed:

1. Double-click on the Server Config icon.

   The Server Config window displays.

2. Click the icon for SQL Server, Backup Server, or Monitor Server in the main Server Config window.
3. Choose the Creating a SQL Server option.

   The SQL Server Name dialog box displays with the default server name in the entry box.

4. Enter a new name and click Continue or press Enter.

How do I give SQL Server a name other than the Windows NT machine name?

By default, SQL Server is created with the same name as the machine on which it exists. To rename SQL Server:

1. Double-click on the Server Config icon.

   The Server Config window displays.

2. Click the icon for SQL Server, Backup Server, or Monitor Server in the main Server Config window.

3. Choose the Creating a SQL Server option.

   The SQL Server Name dialog box displays with the default server name in the entry box.

4. Enter a new name and click Continue or press Enter.

Does including dashes in a Windows NT server name cause conflicts when running on a network that includes UNIX?

No, they are resolved to "#" marks.

Why is the default size for the system procedures database, *sybsystemprocs*, so much larger for SQL Server 11.x than for SQL Server 10.x?

SQL Server 11.x added new system stored procedures. The procedures are compiled during installation and, because of query process differences, they take up more room.

What are the new minimum sizes for the *master*, system procedure, and auditing databases?

The *master* database is 21MB (default).

The *sybsystemprocs* database is 19MB.

The auditing database, *sybsecurity*, remains the same size as in SQL Server 10.x, at 5MB.

See the installation and configuration guides for details about adjusting these sizes for your particular site.

➤ ***Note***

User-created stored procedures require 20 percent disk space in SQL Server 11.x. In addition, provide extra disk space if you intend to add more user-stored procedures. See the information on assigning space and devices to databases in the *System Administration Guide*.

Should I use NTFS or FAT file system for Sybase devices?

The choice depends upon your environment. Generally, Sybase recommends using NTFS. NTFS is more secure than FAT, and has other advantages such as supporting long file names, recoverability, and hot-fixing. You may want to benchmark your applications to determine which would be a better solution for you.

Can I use virtual memory to increase SQL Server memory?

No, SQL Server memory is shared memory implemented on Windows NT as a file mapping object. The file chosen to back the file mapping operation is the system's paging file. Although SQL Server memory and virtual memory are both backed by the system page file, they work differently. Virtual memory is not shared among processes, whereas SQL Server memory is.

Can I have more than one Backup Server or Monitor Server?

Yes, you can have more than one Backup Server, especially if you have remote machines as well as local machines.

No, you can only have one Monitor Server.

Can I use domain login security in place of SQL Server login security?

Yes. For information about managing login security, see *Configuring and Administering SQL Server for Windows NT.*

How can I schedule events to occur at a specific time?

SQL Server 11.x sets events to occur upon startup. The server does not contain a utility, such as **cron**, that executes processes at specified times. You can use a third-party product, such as MKS Toolkit, for UNIX-style utilities.

What do the entries in *sql.ini* stand for?

The SQL interfaces file contains the name and address of every known SQL Server. When you use a client program to connect to a SQL Server, the client program looks up the SQL Server name in the *sql.ini* file and then connects to SQL Server using the specified address.

When would I start SQL Server from the command line?

In most circumstances you use the **services manager** to start SQL Server rather than executing the **sqlsrvr** program directly.

However, when you wish either to read the log as SQL Server starts up, or to choose different parameters for troubleshooting, start the SQL Server from the command line. Start the SQL Server from the command line by executing the **sqlsrvr** program by typing:

```
sqlsrvr -d device_name [-e errorlog_file] [-m]
[-r master_mirror] [-M sharedmem_directory]
[-p sso_login_name] [-g] [-G machine_name]
```

Improving Performance

How can I improve performance during functional suites?

The term "functional suite" refers to functionality testing that you perform on a Sybase product. Memory configuration is critical to performance. SQL Server 11.x requires more default memory allocation.

You may need more memory for queries to run efficiently. The SQL Server kernel itself uses a certain amount of memory. The remaining memory configured for the SQL Server is divided between the procedure cache and the data cache. The key area to look at is the actual number of physical I/Os that are being recorded.

Examine physical I/Os (using **set statistics io**) to determine whether SQL Server is going to the disk more often than is absolutely necessary. Subtract this number from the logical I/O number to get the number of cache hits. If the physical I/O is consistently high, consider increasing the size of data cache.

Other Considerations

Are any Sybase features available only on Windows NT?

SQL Server 11.x for Windows NT offers the same features as other Sybase-supported platforms.

Should I use Windows NT Workstation or Server for SQL Server 11.x?

The Windows NT Server allows multiple network connections. NT Server also is preferred for system administration tasks.

F Create Database Devices on AIX with Additional Space

Allow 1MB for the Logical Volume Control Block (LVCB) when creating SQL Server 11.0.x devices on IBM RISC System/6000 AIX. This appendix describes why and points out a common disk space issue that occurs.

Allow Space for LVCB

When you create a database device with **disk init**, you must allow 1MB on the device for the LVCB, because of the way that AIX and Sybase together manage space.

Factors that Affect Disk Space Allocation

Several factors affect how disk space is allocated:

- SQL Server allocates space for databases in units of 256 pages, or 1/2MB.
- AIX creates logical volumes in physical partitions (PP), typically 4MB.
- The LVCB uses 512 bytes allocated to a logical volume.
- Sybase reserves the first 4K (2 pages) on a database device for the LVCB by setting **vstart** to 2.
- Sybase's **create** and **alter database** commands use whole megabytes only.

When you try to create a database the same size as the logical volume, SQL Server creates a smaller database than you requested because of the reserved space. You must make the logical volume at least 1MB larger than the size of the database you wish to put on it. The example in the next section describes each step and its effect.

Example: Creating a Logical Volume

The table below describes steps in creating a 20MB device.

Table F-1: Creating a device

Step	Description	Action
Create a logical volume.	AIX requires that you create logical volumes, known as physical partitions (PP). Physical partitions must be at least 2MB, but typically are 4MB. For example, you could create a logical volume of 16MB, 20MB, 24MB, and so on.	Create a 20MB logical volume on AIX.
Run **disk init.**	**disk init** offsets the device by 2 pages by automatically setting **vstart** (the device's starting page number) to 2. SQL Server does this to avoid overwriting AIX's Logical Volume Control Block (LVCB).	Subtract 2 pages from the page size of the logical volume. 10,240 pages - 2 =10,238 pages. **disk init** **name = "*device1*",** **physname = "*physname1*",** **vdevno = 5,** **size =10238**
Create the database.	Round down to the nearest 1MB to avoid fragmentation. The **create** and **alter database** commands allow you to specify database size in whole megabytes only.	*device1* is less than 20MB. Therefore, create a 19MB database: **create database *database1* on *device1* = 19**

The Mysterious 1/2MB

In our example, we created a 20MB logical volume, and had less than 20MB for **disk init** because of the LVCB.

We created a 19MB database on the device, **even though disk init will allow more**.

If you create a larger database:

```
create database database2 on device1 = 20
```

Sybase tries to give you all the space possible, so it allocates to the nearest 1/2MB for the database, resulting in 19.5MB. **This causes fragmentation later if you have to re-load the database**: You cannot alter or re-create a 19.5MB database, because the **create** and **alter database** commands only accept whole-megabyte sizes.

For details on avoiding database fragmentation, see TechNote 1324, *Segment Remapping with load database When Moving a Database.*

How *disk init* and *buildmaster* Set Disk Space

On AIX, SQL Server release 11.0.x **disk init** and **buildmaster** set **vstart** to 2 if **vstart** is unspecified or if **vstart** is specified as 0. **vstart** is the starting page for the device. Setting **vstart** to 2 allows Sybase to skip the first 2 pages to avoid overwriting AIX's Logical Volume Control Block (LVCB).

This table shows the effect of SQL Server on the logical volume.

Command	Behavior	Required Action
disk init	Sets **vstart** to 2 if **vstart** is either: Set to 0 Unspecified	Subtract the **vstart** setting from the **disk init size** parameter.
buildmaster	(Used during SQL Server installation) Offsets data in the *master* device by 2 pages.	Subtract the **vstart** setting from the **buildmaster -s** (size) parameter.

Your database device will be smaller than the size of your logical volume by the number of pages set by **vstart**. The default setting is 2.

If you do not adjust the **disk init** size for **vstart**, error 5123 can occur:

```
disk init encountered an error while attempting to
open/create the physical file...
```

➤ ***Note***

See the *Sybase SQL Server System Administration Guide* for information about **disk init** and **vstart.**

G SQL Server 11.x Interoperability and Platform Compatibility

This appendix contains the SQL Server 11.x product interoperability and platform compatibility matrices. Sybase SQL Server 11.x has been extensively tested against other Sybase products. The Sybase products in the matrices have been verified to work with SQL Server 11.x.

Verified Interoperabilities

The following table shows Sybase product versions that work with SQL Server 11.x.

Table G-1: Product interoperability

Products	SQL Server 11.x	Comments
DB-Library™ 4.2.5	X	For complete compatibility information, see the *Open Client/Server Supplement* for your platform.
DB-Library 4.6.2	X	
DB-Library 10.0.2	X	
DB-Library 10.0.3	X	
DB-Library 10.0.4	X	
DB-Library 11.1	X	
Client-Library™ 10.0.2	X	For Distributed Services support, see "Open Client/Server Platform Compatibilities" on page G-3.
Client-Library 10.0.3	X	
Client-Library 10.0.4	X	
Client-Library 11.1	X	
Open Server™ 10.0.2	X	To move from SQL Server 4.9.2, you need to upgrade Open Server applications to Open Server 10.0.3 or higher.
Open Server 10.0.3	X	
Open Server 10.0.4	X	
Open Server 11.1	X	

Table G-1: Product interoperability (continued)

Products	SQL Server 11.x	Comments
Embedded SQL™/C 10.0.3	X	no comments
Embedded SQL/C 10.0.4	X	
Embedded SQL/C 11.1	X	
Embedded SQL/Cobol 10.0.3	X	
Embedded SQL/Cobol 10.0.4	X	
Embedded SQL/Cobol 11.1	X	
XA Server™ for Tuxedo	X	no comments
XA Library™	X	
Replication Server® 10.0.3	X	no comments
Replication Server 10.1	X	
Replication Server 11.0.1	X	
SQL Server Manager™ 10.0.1		SQL Server Manager 11.0 is required for SQL Server 11.x. They are bundled together.
SQL Server Manager 11.0	X	
SQL Server Monitor™ 10.0.1	X	SQL Monitor 11.0 is required for SQL Server 11.x.
SQL Server Monitor 11.0	X	
SQL Debug®10.0.3		SQL Debug does not support SQL Server 11.x.
SA Companion 10.0		SQL Server 11.x does not support SA Companion. However, SQL Server Manager is bundled with 11.x.
OmniCONNECT™ 10.1.2	X	no comments
OmniCONNECT 10.5	X	
IQ™ 11.0.3	X	IQ does not support pre-11.0.1 SQL Server releases as a catalog server. However, attached database servers can include SQL Server releases 4.9.1 or higher.
PowerBuilder® 4	X	no comments

Open Client/Server Platform Compatibilities

This section shows the platforms combinations for which Open Client./Server and SQL Server have been tested as follows:

Open Client/Server 10.0.3 and 11.1

This table shows Open Client/Server (OCS) 10.0.3, 11.1 and SQL Server 4.9.2, 10.0.2.x, 11.1 with the platform combinations that have been tested. Blank cells indicate combinations that have not been tested.

Table G-2: OCS compability

Platform/ Operating System	SQL Server			Client-Library		DB-Library		Open Server	
	11.x	10.02x	4.92	11.1	10.03	11.1	10.03	11.1	10.03
Digital UNIX 3.2	X	X[a]							
HP 9000/800 HP-UX 10.0.1 (SQL Server on HP-UX 10.0, OCS on HP-UX 10.0.1)	X		X	X	X	X	X	X	X
IBM RS/6000 AIX 4.1.4 (SQL Server on AIX 4.1.3, OCS on AIX 4.1.4)	X	X		X		X		X	
Solaris 2.4 SPARC	X	X	X	X	X	X	X	X	X
Windows 3.1	n/a	n/a	n/a	X	X	X	X	n/a	n/a
Windows NT 3.5 (SQL Server on NT 3.5, OCS on NT 3.5.1)	X	X		X	X	X		X	X
Windows 95 4.0	n/a	n/a	n/a	X	X	X		X	X

a. With symmetrical multprocessing (SMP) systems only.

Distributed Services Platforms

The following table shows the Open Client/Server platform support for Distributed Services. These services are optional for Release 11.1.

Existing applications work the same as for Release 10.x without them.

The abbreviations are:

- C – clients only
- S – servers only
- CS – both clients and servers

Table G-3: Distributed Services support

Platform	DCE	CyberSafe Challenger	Microsoft NT Registry	Novell Netware 4.0	Banyan StreetTalk
HP 9000/800 HP-UX 10.0.1	CS	CS			
IBM RS/6000 AIX 4.1.4	CS				
Sun Solaris 2.4 (SPARC)	CS	CS			
Windows 3.1		C	C	C	C
Windows 95 4.0			CS		
Windows NT 3.5			CS		

Security Guardian

Client applications require Client-Library 11.1 to use Security Guardian, a separate product that allows SQL Server to use third-party security services. One Security Guardian is required for each SQL Server, and it must reside on the same node/system.

Security Guardian is available on these platforms:

- HP 9000/800 HP-UX 10.0.1
- IBM RS/6000 AIX 4.1.4
- Sun Solaris 2.4 (SPARC)
- Windows NT 3.5

Open Client/Server 10.0.3 and 10.0.4

This table shows OCS 10.0.3, 10.0.4 and SQL Server 4.9.2, 10.2.x, 11.x with platform combinations that have been tested. Blank cells indicate combinations that have not been tested.

Table G-4: OCS and SQL Server platform combinations

Platform/ Operating System	SQL Server			Client-Library		DB-Library		Open Server	
	11.x	10.02x	4.92	10.03	10.04	10.03	10.04	10.03	10.04
Digital UNIX 3.2	X	X[a]			X		X		X
HP 9000/800 HP-UX 10.0.1 (SQL Server on HP-UX 10.0, OCS on HP-UX 10.0.1)	X		X	X	X	X	X	X	X
IBM RS/6000 AIX 4.1.4 (SQL Server on AIX 4.1.3, OCS on AIX 4.1.4)	X	X		X		X	X	X	
Solaris 2.4 SPARC	X	X	X	X	X	X	X	X	X
SunOS 4.1.3	n/a			X	X		X		X
Windows 3.1	n/a	n/a	n/a	X	X	X	X	n/a	n/a
Windows NT 3.5 (SQL Server on NT 3.5, OCS on NT 3.5.1)	X	X	n/a	X	X	X	X	X	X
Windows 95 4.0	n/a	n/a	n/a	X	X	X	X	X	X

a. With symmetrical multiprocessing (SMP) only.

Replication Server 11.0.1

Replication Server 11.0.1 has been tested with Open Client/Server and SQL Server for these versions and platforms:.

Table G-5: Replication Server compatibility

Platform/ Operating System	SQL Server			Client-Library		DB-Library		Open Server	
	11.x	10.02x	4.92	11.1	10.03	11.1	10.03	11.1	10.03
HP 9000/800 HP-UX 10.0.1	X		X	X	X	X	X	X	X
Solaris 2.4 SPARC	X	X	X	X	X	X	X	X	X

Table G-5: Replication Server compatibility (continued)

Platform/ Operating System	SQL Server			Client-Library		DB-Library		Open Server	
	11.x	10.02x	4.92	11.1	10.03	11.1	10.03	11.1	10.03
Windows NT 3.5	X	X		X	X	X		X	X

H ANSI SQL '89 and FIPS 127-1 Standards Compliance

As of release 10.0, the following changes bring SQL Server into full compliance with both the ANSI SQL '89 and FIPS 127-1 standards:

Table H-1: Release 10.0 standards compliance

Feature	Summary	For more information
Transactions	ANSI standardization and release enhancements impact the behavior of chained and serializable transactions. For example, the **set** command enables you to set chained on / off and set isolation levels.	*Reference Manual* *Transact-SQL User's Guide*
IDENTITY columns and *@@identity* global variable	SQL Server guarantees that a column defined as IDENTITY contains a unique value within the entire table when a row is inserted.	*Reference Manual* *Transact-SQL User's Guide*
Database cursors	Cursors support the processing of a row at a time in Transact-SQL.	*Reference Manual* *Transact-SQL User's Guide*
Integrity Enhancement Feature (IEF)	This feature provides for a declarative approach to integrity, compared to SQL Server. SQL Server 11.x supports both approaches: **ANSI '89** / **SQL Server** declare unique constraint / create unique index declare referential integrity / create trigger declare table check constraint / create trigger declare column check constraint / create trigger declare column default value / create and bind default declare **not null** constraint / **not null** is the default The advantages of each are as follows: • You can name, define once and reuse SQL Server rules and defaults. Triggers can support business rules in the server. • ANSI '89 is an accepted industry standard. All integrity constraints are defined and accessed in one place.	About SQL Server triggers, rules, and defaults: *Reference Manual* *Transact-SQL User's Guide*
Schema	This new **create schema** command creates a new collection of tables, views, and permissions in the current database. All objects in a **create schema** statement can be committed and rolled back as one unit.	*Reference Manual*
Views	New features include: • **distinct** clause to prohibit duplicate rows in a view • **with check option** clause to check updates statements against the view's selection criteria	*Reference Manual*

Table H-1: Release 10.0 standards compliance (continued)

Feature	Summary	For more information
like predicate and **escape** clauses	ANSI pattern matching characters, such as @ and %, can appear in a string as actual occurrences of the character, rather than the pattern matching character.	*Reference Manual* (see information on wildcard characters) *Transact-SQL User's Guide* (see information on matching character strings)
Federal Information Processing 127-1 Standard (FIPS)	The **set fipsflagger** option issues an informational message if Transact-SQL statements in an application do not conform to the FIPS standard.	*Reference Manual*

Index

D

E

F

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W