SYBASE®

What's New in Sybase® Adaptive Server™ Enterprise

Adaptive Server™

Document ID: 36440-01-1150-02

September 1997

Copyright Information

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

Table of Contents

About This Book

1. New Features in Adaptive Server Release 11.5

2. System Changes in Adaptive Server Release 11.5

3. New Features and System Changes in SQL Server Release 11.0

4. New Features and System Changes in SQL Server Release 10.0

Index

List of Figures

List of Tables

About This Book

What's New in Adaptive Server Enterprise? is an introduction to the new Sybase® Adaptive Server™ Enterprise features and the commands, system procedures, system tables, and documentation that support them.

This document also covers the new features added and system changes made in releases 10.0 and 11.0. This is provided for users who are upgrading to release 11.5 from release 4.9.x or 10.0.x.

Audience

This manual is intended for customers who are upgrading to Adaptive Server 11.5 from Sybase SQL Server™.

How to Use This Book

Table 1 lists the chapters you should read and **the order in which you should read them** before upgrading to Adaptive Server release 11.5.

Table 1: How to use this book

If Upgrading from This Server Release	Read These Chapters
11.0 or 11.0.x	• Chapter 1, "New Features in Adaptive Server Release 11.5" • Chapter 2, "System Changes in Adaptive Server Release 11.5"
10.0, 10.0.x, or 10.1	• Chapter 3, "New Features and System Changes in SQL Server Release 11.0" • Chapter 1, "New Features in Adaptive Server Release 11.5" • Chapter 2, "System Changes in Adaptive Server Release 11.5"

Table 1: How to use this book (continued)

If Upgrading from This Server Release	Read These Chapters
4.9.x	• Chapter 4, "New Features and System Changes in SQL Server Release 10.0" • Chapter 3, "New Features and System Changes in SQL Server Release 11.0" • Chapter 1, "New Features in Adaptive Server Release 11.5" • Chapter 2, "System Changes in Adaptive Server Release 11.5"

This manual consists of the following chapters:

- Chapter 1, "New Features in Adaptive Server Release 11.5," describes the new features added to release 11.5.
- Chapter 2, "System Changes in Adaptive Server Release 11.5," describes the Adaptive Server system changes made to support the new 11.5 features. Also discussed is how these additions and changes might affect your existing applications.
- Chapter 3, "New Features and System Changes in SQL Server Release 11.0," describes the features added and system changes made in release 11.0. Also discussed is how these additions and changes might affect your existing applications.
- Chapter 4, "New Features and System Changes in SQL Server Release 10.0," describes the features added and system changes made in release 10.0. Also discussed is how these additions and changes might affect your existing applications.

Adaptive Server Enterprise Documents

The following documents comprise the Sybase Adaptive Server Enterprise documentation:

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

  A more recent version of the *Release Bulletin* may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use SyBooks™-on-the-Web.

- The Adaptive Server installation documentation for your platform – describes installation and upgrade procedures for all Adaptive Server and related Sybase products.
- The Adaptive Server configuration documentation for your platform – describes configuring a server, creating network connections, configuring for optional functionality, such as auditing, installing most optional system databases, and performing operating system administration tasks.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server release 11.5, the system changes added to support those features, and the changes that may affect your existing applications.
- *Navigating the Documentation for Adaptive Server* – an electronic interface for using Adaptive Server. This online document provides links to the concepts and syntax in the documentation that are relevant to each task.
- *Transact-SQL User's Guide* – documents Transact-SQL®, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the *pubs2* and *pubs3* sample databases.
- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources and user and system databases, and specifying character conversion, international language, and sort order settings.
- *Adaptive Server Reference Manual* – contains detailed information about all Transact-SQL commands, functions, procedures, and datatypes. This manual also contains a list of the Transact-SQL reserved words and definitions of system tables.
- *Performance and Tuning Guide* – explains how to tune Adaptive Server for maximum performance. This manual includes information about database design issues that affect performance, query optimization, how to tune Adaptive Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.
- The *Utility Programs* manual for your platform – documents the Adaptive Server utility programs, such as **isql** and **bcp**, which are executed at the operating system level.

- *Security Administration Guide* – explains how to use the security features provided by Adaptive Server to control user access to data. This manual includes information about how to add users to Adaptive Server, administer both system and user-defined roles, grant database access to users, and manage remote Adaptive Servers.
- *Security Features User's Guide* – provides instructions and guidelines for using the security options provided in Adaptive Server from the perspective of the non-administrative user.
- *Error Messages* and *Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Component Integration Services User's Guide for Adaptive Server Enterprise and OmniConnect* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Adaptive Server Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Master Index for Adaptive Server Publications* – combines the indexes of the *Adaptive Server Reference Manual, Component Integration Services User's Guide, Performance and Tuning Guide, Security Administration Guide, Security Features User's Guide, System Administration Guide,* and *Transact-SQL User's Guide*.

Other Sources of Information

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

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

  Refer to *Installing SyBooks* in your documentation package for instructions on installing and starting SyBooks.
- SyBooks-on-the-Web is an HTML version of SyBooks that you can access using a standard Web browser.

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

Conventions

The following style conventions are used in this manual:

- In a sample screen display, commands you should enter exactly as shown are given in:

  `this font`

- In a sample screen display, words which you should replace with the appropriate value for your installation are shown in:

  `this font`

- In the regular text of this document, the names of files and directories appear in italics:

  /usr/u/sybase

- The names of programs, utilities, procedures, and commands appear in bold type:

  sybsetup

- Commands for both the C and Bourne shells are provided in this document when they differ. The initialization file for the C shell is called *.cshrc*. The initialization file for the Bourne shell is called *.profile*. If you are using a different shell, such as the Korn shell, refer to your shell-specific documentation for the correct command syntax.

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

Table 2: SQL syntax conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords are in **bold**.
variable	Variables, or words that stand for values that you fill in, are in *italics*.
{ }	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
[ ]	Brackets mean choosing one or more of the enclosed options is optional. Do not include brackets in your option.
( )	Parentheses are to be typed as part of the command.
|	The vertical bar means you can select only one of the options shown.

Table 2: SQL syntax conventions (continued)

Key	Definition
,	The comma means you can choose as many of the options shown as you like, separating your choices with commas to be typed as part of the command.

If You Need Help

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

1 New Features in Adaptive Server Release 11.5

This chapter describes the new features and system changes introduced in Sybase® Adaptive Server™ Enterprise release 11.5.

In addition to providing many new server features, Adaptive Server 11.5 provides a more comprehensive product by incorporating previous products and tools:

- SQL Server Manager™ has been replaced by Sybase Central™

  For more information, see "Sybase Central" on page 1-30.
- OmniConnect™ has been replaced by Component Integration Services

  Many of the OmniConnect release 10.5 features have been incorporated into the Component Integration Services (CIS) feature of Adaptive Server 11.5. For more information on CIS, see "Component Integration Services" on page 1-6.

New Features in Release 11.5

The new features in Adaptive Server release 11.5 are:

Asynchronous Prefetch

Asynchronous prefetch improves performance by anticipating the pages that will be required for certain well-defined classes of database activities whose access patterns are predictable. The I/O requests for these pages are issued before the query needs them so that most pages are in cache by the time query processing needs to access the page.

Asynchronous prefetch can improve performance for:

- Sequential scans, such as table scans, clustered index scans, and covered nonclustered index scans
- The **update statistics** command
- Accesses via nonclustered indexes
- Some **dbcc** checks
- Recovery

Asynchronous prefetch can improve the performance of queries that access large numbers of pages, such as decision support system (DSS) applications, as long as the I/O subsystems on the machine are not saturated.

Asynchronous prefetch cannot help (or may help only slightly) when the I/O subsystem is already saturated or when Adaptive Server is CPU-bound. It can be used in some online transaction processing (OLTP) applications, but to a much lesser degree, since the queries in OLTP applications usually perform fewer I/O operations.

➤ ***Note***

The default setting for **global async prefetch limit** is 10, meaning that asynchronous prefetch is enabled when you install or upgrade to release 11.5, and that all pools can use up to 10 percent of the buffers for asynchronous reads.

Table 1-1 shows where to find additional information on asynchronous prefetch.

Table 1-1: Additional information on asynchronous prefetch

For information On	See
How queries can benefit from asyncronous prefetch	Chapter 18, "Tuning Asynchronous Prefetch," in the *Performance and Tuning Guide*
Configuring pool limits	Chapter 9, "Configuring Data Caches," in the *System Administration Guide*
Configuring server-wide limits	Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*
Tuning asynchronous prefetch limits	Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide*

Auditing Enhancements

The auditing system provides the following enhancements:

- Support of multiple audit tables. This capability allows you to archive and process audit data with no manual intervention and no loss of audit records.
- A new, streamlined, user interface for the System Security Officer to control auditing. In the new interface, which replaces the current auditing interface, a single system procedure, **sp_audit**, provides the capability to set all auditing options.
- The addition of events that can be audited, including:
  - Creation of objects

- Binding and unbinding of rules, defaults, and messages
- All actions by individual users
- All actions performed with a particular role active
- A set of server-wide, security-relevant events

New System Procedures

Use the system procedures listed in Table 1-2 to manage the auditing process.

Table 1-2: New system procedures for auditing

System Procedure	Description
sp_addauditrecord	Adds user-defined audit records (comments) into the audit trail. Users can add these records only if a System Security Officer enables ad hoc auditing with **sp_audit**.
sp_audit	Enables and disables all auditing options. This is the only system procedure required to establish the events to be audited.
sp_configure	
"auditing"	Enables or disables auditing for Adaptive Server. The parameter takes effect immediately upon execution of **sp_configure**. Auditing occurs only when this parameter is enabled.
"audit queue size"	Establishes the size of the audit queue. Because this parameter affects memory allocation, it does not take effect until Adaptive Server is restarted.
"current audit table"	Sets the current audit table. This takes effect immediately upon execution of **sp_configure**.
"suspend auditing when full"	Controls the behavior of the audit process when an audit device becomes full. This takes effect immediately upon execution of **sp_configure**.
sp_displayaudit	Displays the active auditing options.

➤ ***Note***

The system procedures **sp_auditoption**, **sp_auditdatabase**, **sp_auditobject**, **sp_auditsproc**, and **sp_auditlogin** are obsolete. Use **sp_audit** for setting the options that you set with those system procedures in pre-11.5 releases.

Table 1-3 shows the system procedures that were used to set auditing options in pre-11.5 releases and the types of auditing options that replace them in Adaptive Server release 11.5. The auditing options in the groups mentioned in the second column of Table 1-3 are described in the discussion of **sp_audit** in the *Adaptive Server Reference Manual*.

Table 1-3: Comparison of pre-11.5 and 11.5 options

Option set in pre-11.5 releases	Option to set In 11.5
sp_auditoption	"global" auditing options
sp_auditdatabase	"database-specific" auditing options
sp_auditobject	"object-specific" auditing options
sp_auditsproc	"exec" auditing options
sp_auditlogin	"user-specific" auditing options

For example, in pre-11.5 releases, you used the following command to audit the actions of deleting, updating and inserting information in the *authors* table of the *pubs2* database:

```
sp_auditobject "authors", "pubs2", "both", "dui"
```

In release 11.5, these actions are covered by the **delete, update**, and **insert** options in the "object-specific" group. You use the following commands to achieve the same results as you did using the preceding, obsolete commands:

```
sp_audit "delete", "all", "authors", "on"
sp_audit "update", "all", "authors", "on"
sp_audit "insert", "all", "authors", "on"
```

To install auditing, see the Adaptive Server configuration guide for your platform. For more information about auditing, see Chapter 8, "Auditing," in the *Security Administration Guide*.

case Expression

Using **case** expression simplifies standard SQL expressions by allowing a **when...then** construct instead of an **if...else** construct. A **case** expression is more concise than an **if...else** statement and Adaptive Server's response time is faster, since the optimizer processes fewer lines of SQL code. Also, queries written using **case** expressions are often easier to read and understand than those written in standard SQL. Because of the concise nature of **case** expressions, they are allowed anywhere value expressions are used.

A **case** expression is written using the following syntax:

```
case
        [when search_condition1 then result1]
        [when search_conditionn then resultn]
end
```

For more information about the syntax of **case** expressions, see the *Adaptive Server Reference Manual.*

For more information and examples of **case** expressions, see the *Transact-SQL User's Guide*.

Component Integration Services

The Component Integration Services (CIS) feature allows you to connect to remote Sybase and non-Sybase databases. It presents a uniform view of enterprise data to client applications and provides location transparency to the data sources that you access.

When you access both Sybase and non-Sybase databases on different servers, you can use the host data files, tables, triggers, views, remote procedure calls, and most commands in database systems such as Adaptive Server, Informix, DB2, and Oracle.

Component Integration Services can be used by anyone who needs to access multiple data sources or legacy data. It can also be used by anyone who needs to migrate data from one server to another.

A single server is often used to access data on multiple external servers. Component Integration Services manages the data

regardless of the location of the external servers. Data management is transparent to the client application.

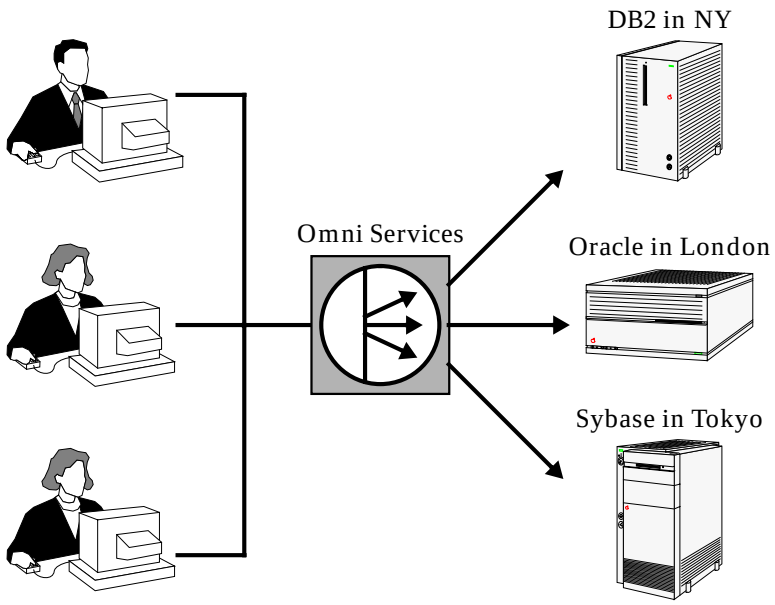

Figure 1-1: CIS connects to multiple vendor databases

Using Component Integration Services, you can:

- Access tables in remote servers as if the tables were local.
- Perform joins between tables in multiple, remote, and heterogeneous servers. For example, you can join tables between Adaptive Server and an Oracle DBMS, between Adaptive Server and Record Management System (RMS) files in OpenVMS, and between tables in multiple Adaptive Servers.
- Use the **select into** command to transfer the contents of one table into a new table on any supported remote server.
- Provide applications such as PowerBuilder®, Microsoft Access, and DataEase with transparent access to heterogeneous data.
- Maintain referential integrity across heterogeneous data sources.
- Access native remote server capabilities using the Component Integration Services passthrough mode.

See “Differences Between CIS and OmniConnect” on page 2-3 for information on the differences between OmniConnect release 10.5 and the Component Integration Services feature incorporated into Adaptive Server 11.5.

For more information on using Component Integration Services, see the *Component Integration Services User's Guide*.

New Commands

Table 1-4 lists the new commands that support the Component Integration Services feature:

Table 1-4: Commands for CIS

Command	Description
connect to...disconnect	Connects to the specified server and disconnects the connected server.
create existing table	Creates a proxy table, then retrieves and stores metadata from a remote table, and places the data into the proxy table.
set cis_rpc_handling	Determines whether Component Integration Services handles outbound remote procedure call (RPC) requests by default.
set transactional_rpc	Controls the handling of remote procedure calls.

New System Procedures

Table 1-5 lists the new system procedures that support the Component Integration Services feature.

Table 1-5: New system procedures for CIS

System Procedure	Description
sp_addexternlogin	Creates an alternate login account and password to use when communicating with a remote server through CIS.
sp_addobjectdef	Specifies the mapping between a local table and an external storage location. **sp_addobjectdef** replaces the **sp_addtabledef** command.

Table 1-5: New system procedures for CIS (continued)

System Procedure	Description
sp_autoconnect	Defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.
sp_configure "enable cis"	Enables or disables CIS. Default is 0 (disabled).
sp_defaultloc	Defines a default storage location for objects in a local database.
sp_dropexternlogin	Drops the definition of a remote login previously defined by **sp_addexternlogin**.
sp_dropobjectdef	Deletes the external storage mapping provided for a local object.
sp_passthru	Allows the user to pass a SQL command buffer to a remote server.
sp_remotesql	Establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client.

Modified System Procedures

The following system procedures have been modified to support the Component Integration Services feature:

- **sp_addserver**
- **sp_dropserver**
- **sp_help**
- **sp_helpdb**
- **sp_helpserver**
- **sp_serveroption**

create index Enhancements

In earlier releases of SQL Server™, you could use the **create clustered index...with sorted_data** to skip the sort step on data that was already in sorted order. However, **create clustered index** always copied the data pages to a new location on the data devices.

In release 11.5, **create clustered index...with sorted_data** does not sort or copy data pages except when certain optional clauses are specified to **create index**. If these clauses are specified for partitioned tables, a parallel sort must be performed.

See Chapter 23, "Maintenance Activities and Performance," in the *Performance and Tuning Guide* for more information.

dbcc Enhancements

dbcc has been enhanced to improve its performance in parallel environments and to expand the way in which it detects, records, and reports errors. The **dbcc checkstorage** command allows you to check a database that is in use, with little or no impact on performance. It stores the results of the check in the *dbccdb* database. When the check is complete, you can use the new **dbcc** stored procedures to generate reports based on the data that is collected.

Advantages of Using *dbcc checkstorage*

The **dbcc checkstorage** command:

- Uses a named cache so that running it on the target database does not interfere with concurrent use of that database
- Scales linearly with the aggregate I/O throughput
- Does not lock tables or pages for extended periods, which allows **dbcc checkstorage** to locate errors while allowing concurrent update activity
- Combines many of the checks provided by the other **dbcc** commands
- Provides a detailed description of space usage in the target database
- Separates the functions of checking and reporting, which allows custom evaluation and report generation
- Records **dbcc checkstorage** activity and results in the *dbccdb* database, which allows trend analysis and a source of accurate diagnostic information for support

See Chapter 18, "Checking Database Consistency," in the *System Administration Guide* for more information on **dbcc**.

System and Stored Procedures for Creating *dbccdb*

Before you can use **dbcc checkstorage**, you must set up the *dbccdb* database. Table 1-6 lists the system and stored procedures for creating *dbccdb*. **sp_plan_dbccdb** is the only **system** procedure. It resides in *sybsytemprocs*. **sp_dbcc_createws** and **sp_dbcc_updateconfig** are **dbcc stored** procedures. They reside in *dbccdb*.

Table 1-6: System and stored procedures for creating dbccdb

Procedure	Description
sp_dbcc_createws	Creates a workspace of the specified type and size on the specified segment and database.
sp_dbcc_updateconfig	Updates the configuration parameters for the specified database.
sp_plan_dbccdb	Recommends values for the size of *dbccdb*, suitable devices, sizes for the *scan* and *text* workspaces, cache size, and the number of worker processes for optimal operation.

Stored Procedures for Maintaining *dbccdb*

Table 1-7 lists the **dbcc** stored procedures for maintaining *dbccdb*.

Table 1-7: Stored procedures for maintaining dbccdb

Stored Procedure	Description
sp_dbcc_alterws	Changes the size of the specified workspace to the specified value, and initializes the workspace.
sp_dbcc_deletedb	Deletes all information on the specified database from *dbccdb*.
sp_dbcc_deletehistory	Deletes results of **dbcc checkstorage** operations on the specified database from *dbccdb*.
sp_dbcc_evaluatedb	Recommends values for configuration parameters, using the results from previous **dbcc** operations.
sp_dbcc_runcheck	Runs **dbcc checkstorage** on the specified database and generates a summary report or the report you specify.

Stored Procedures for Generating Reports on *dbccdb*

Table 1-8 lists the **dbcc** stored procedures for generating reports on *dbccdb*.

Table 1-8: Stored procedures for generating reports on dbccdb

Stored Procedure	Description
sp_dbcc_configreport	Reports the configuration parameters for the specified database.
sp_dbcc_differentialreport	Compares the results of **dbcc checkstorage** operations completed on *date1* and *date2* for *dbname..object_name*.
sp_dbcc_faultreport	Reports faults that were recorded for *dbname..object_name* on or before the specified date.
sp_dbcc_fullreport	Reports configuration, statistics, and fault information for *database..object_name* on or before the specified date.
sp_dbcc_statisticsreport	Reports statistics information from the *dbcc_counters* table generated by the **dbcc checkstorage** operation on or before the specified date.
sp_dbcc_summaryreport	Reports all **dbcc checkstorage** operations completed for *dbname* on or before the specified date.

Descending Index Scan Optimization

Descending index scan optimization is a performance enhancement that can improve the performance of queries that use the **desc** keyword in **order by** queries to return result sets in descending order. In earlier releases of SQL Server, descending result sets required a worktable and a sort. In Adaptive Server 11.5, the optimizer can choose to use the index and avoid the sort step, if the strategy reduces the query cost.

Descending index scans can speed the use of both clustered and nonclustered index access, reduce the *tempdb* space required for temporary tables, save the CPU time required for sorts, and shorten the time that locks are held, if descending scans use **holdlock** or transaction isolation level 3. However, there can be an increased chance of deadlocking in some applications.

See Chapter 11, “Setting Configuration Parameters,” in the *System Administration Guide* for information on the **allow backward scans**

configuration parameter, which can be used to disable this optimization if deadlocking is a problem. Descending scans are enabled by default.

This feature does not change the syntax for the **order by** clause; it only changes the way that **order by** clauses with the **desc** keyword can be optimized.

Directory Services

Before Adaptive Server and a client application can communicate with each other, each must know where the other resides on the network. Prior to Adaptive Server 11.5, the only location for storing network service information (server name, service type, and network protocol, device, and address) was the SQL Server interfaces file.

Adaptive Server 11.5 provides an alternative to using the traditional interfaces file by supporting the ability to connect to network service information through third-party directory service providers.

The advantage of using a directory service is that when your network is changed (for example, when a new server is added or a server address is changed), you do not need to update multiple interfaces files. You need to update only the directory service being used by the servers and clients.

Table 1-9 shows the directory service provider and the platform it supports in Adaptive Server 11.5.

Table 1-9: Directory service provider

Directory Service Provider	Platform
Windows Registry	Microsoft Windows NT

If Adaptive Server is not configured to use a directory service provider, it defaults to the Sybase interfaces file for obtaining network service information.

For information on how to configure Adaptive Server for directory services, see the Adaptive Server configuration guide for your platform.

Engine Affinity and Execution Precedence

The new Logical Process Manager in release 11.5 gives you a measure of control over the order in which Adaptive Server executes requests from connections and sessions. Two new capabilities, engine affinity and execution precedence, offer system procedures that let you tell Adaptive Server the performance trade-offs you want to make among individual connections and sessions in mixed work-load environments.

For example, you can rank a client application, login, or stored procedure as preferred with respect to others using system procedures to assign **execution attributes**. Adaptive Server will consider your execution assignments when it places the entity in one of three priority run queues. In addition, you can use **engine affinity** to make suggestions about how to partition Adaptive Server engine resources among connections and sessions.

In a multiple-application environment, establishing execution precedence can improve performance for selected applications and "on the fly" for some running Adaptive Server processes.

You establish which client applications, logins, and stored procedures receive execution precedence when you create a hierarchy among **execution objects** (client applications, logins, and stored procedures) using a new set of Adaptive Server system procedures (see Table 1-10). Adaptive Server uses the hierarchy as a guideline for distributing processing resources among execution objects.

For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.

New System Procedures

You implement and manage execution hierarchy for client applications, logins, and stored procedures using the five categories of system procedures listed in Table 1-10.

Table 1-10: New system procedures for managing execution precedence

Category	System Procedure	Description
User-defined execution class	• **sp_addexeclass** • **sp_dropexeclass**	Creates or drops a user-defined class with custom attributes or changes the attributes of an existing class.

Table 1-10: New system procedures for managing execution precedence

Category	System Procedure	Description
Execution class binding	• **sp_bindexeclass** • **sp_unbindexeclass**	Binds or unbinds both predefined and user-defined classes to client applications and logins.
"On the fly"	• **sp_setpsexe** • **sp_clearpsexe**	Sets or clears attributes of an active session only.
Engines	• **sp_addengine** • **sp_dropengine**	Adds engines to or drops engines from engine groups; creates and drops engine groups.
Reporting	• **sp_showcontrolinfo** • **sp_showexeclass** • **sp_showpsexe**	Reports on engine group assignments, application bindings, and execution class attributes.

Expanded Error Logging

Users can now exercise control over whether certain messages are always logged to the Adaptive Server error log.

By setting the **log audit logon success** or **log audit logon failure** configuration parameter, you can enable or disable the logging of successful or unsuccessful Adaptive Server logins.

You can also specify that a particular message, identified by its message identifier, should be logged, using the **with_log** parameter to the **sp_addmessage** or **sp_altermessage** system procedure. See the *Adaptive Server Reference Manual* for information about these system procedures.

These features affect logging to the Adaptive Server error log on all platforms, as well as logging to the Windows NT Event Log on Windows NT servers, if **event logging** is enabled.

New System Procedures

Table 1-11 lists the new system procedures that support the expanded error logging feature.

Table 1-11: New system procedures for expanded error logging

System Procedure	Description
sp_altermessage	Enables and disables the logging of a system-defined or user-defined message in the Adaptive Server error log.
sp_configure	
"log audit logon success"	Specifies whether to log successful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.
"log audit logon failure"	Specifies whether to log unsuccessful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled

Modified System Procedures

The following system procedures have been modified to support the expanded error logging feature by using the **with_log** parameter:

- **sp_addmessage**
- **sp_addserver**

Extended Stored Procedures (ESPs)

This release supports both user-defined and system-defined extended stored procedures (ESPs).

ESPs provide a method for calling procedural language functions from within Adaptive Server. The procedural language in which the functions are written must be capable of calling C language functions and manipulating C data types.

ESPs allow Adaptive Server to perform a task outside Adaptive Server in response to an event occurring within Adaptive Server. For example, a procedural task outside the relational database management system (RDBMS), such as selling a security, could be invoked by a trigger on a database table that fires when the value of

the security reaches a certain value. Or an email notification or a network-wide broadcast could be sent in response to an event occurring within the RDBMS.

The interface to ESPs is similar to the interface to system procedures and user-defined stored procedures. The difference is that an ESP executes procedural language code, rather than Transact-SQL statements.

XP Server

Extended stored procedures are implemented by an Open Server™ application called XP Server, which runs on the same machine as Adaptive Server. Running ESPs in a separate process protects Adaptive Server from failing because of faulty ESP code. Adaptive Server and XP Server communicate through remote procedure calls (RPCs).

The function that implements the ESP is compiled and linked into a dynamic link library (DLL) or a shared library. Adaptive Server looks in the system tables for the function that has the same name as the requested ESP and passes the function name and the DLL name to XP Server.

XP Server is responsible for:

- Loading the DLL, if it is not already loaded.
- Invoking the function that implements the ESP.
- Passing the function's return status, output parameters, and results back to Adaptive Server.

Adaptive Server passes the results of the function to the client.

Adaptive Server starts XP Server on the first ESP request and shuts down XP Server when Adaptive Server shuts down.

The installation or configuration program for your platform adds the default *<hostname>_XP* entry to the interfaces file and the *sysservers* table. Additional XP Server entries must be added to the interfaces file using **dsedit**. Additional XP Server entries to the *sysservers* table must be manually added. There is no *run_server* file associated with XP Server.

See the *Transact-SQL User's Guide* for detailed information about using and developing your own ESPs.

To develop your own ESPs, you need to purchase the Open Server product.

System-Defined ESPs

Table 1-12 lists the system-defined ESPs.

Table 1-12: System-defined ESPs

System Procedure	Description
xp_cmdshell	Executes a native operating system command on the host system running Adaptive Server.
xp_enumgroups	Displays groups for a specific Windows NT domain. (Windows NT only)

In addition, several Windows NT-specific ESPs provide Windows NT features, such as event logging and Adaptive Server mail support.

See Chapter 5, "System Extended Stored Procedures," in the *Adaptive Server Reference Manual* for a complete list of system ESPs.

New System Procedures for ESPs

Table 1-13 lists the system procedures that support ESPs.

Table 1-13: New system procedures for ESP

System Procedure	Description
sp_addextendedproc	Creates an ESP.
sp_configure	
"esp execution priority"	Sets the priority of the XP Server thread for ESP execution.
"esp execution stacksize"	Sets the size of the stack, in bytes, to allocate for ESP execution.
"esp unload dll"	Specifies whether DLLs that support ESPs should automatically be unloaded from XP Server memory after the ESP call has completed.
"xp_cmdshell_context"	Sets the security context for the operating system command to be executed using the **xp_cmdshell** system ESP.
sp_dropextendedproc	Drops an ESP.
sp_freedll	Unloads a DLL that was loaded into XP Server memory to support the execution of an ESP.

Table 1-13: New system procedures for ESP (continued)

System Procedure	Description
sp_helpextendedproc	Displays ESPs in the current database with their associated DLL files.

Supporting Commands

The following Transact-SQL commands have been enhanced to support ESPs:

- **create procedure**
- **drop procedure**
- **execute**

Increased Referential Integrity Limits

The referential integrity relaxed limit feature increases the maximum number of table references allowed in a query from 16 to 192. The new limit eliminates the need to write special triggers to compensate for the previous 16-table reference limitation.

When a table has references to other tables, Adaptive Server scans the referenced tables when the table is being modified. Adaptive Server uses **scan descriptors** to manage the table scans. A scan descriptor is an internal data structure that manages the scan of the tables being queried. To make efficient use of Adaptive Server resources, you can configure Adaptive Server to use an appropriate number of scan descriptors for your system.

For more information on using referential integrity, see the *Transact-SQL User's Guide*.

For information on configuring scan descriptors, see the **number of aux scan descriptors** configuration parameter in Chapter 11, "Setting Configuration Parameters," of the *System Administration Guide*.

Metadata Cache Management

Adaptive Server allows you to manage individual metadata caches for:

- User indexes

- Objects such as procedures, triggers, views, rules, and defaults
- Databases

Managing individual metadata caches for these objects is beneficial for a server that contains a large number of user indexes and objects and where there is high concurrency among users.

A **metadata cache** is a reserved area of memory for tracking information for indexes, objects, or databases. When you configure a metadata cache for indexes, objects, or databases, Adaptive Server accesses the information directly from the in-memory structure that describes them in the *sysindexes*, *sysobjects*, or *sysdatabases* row.

This method improves performance because Adaptive Server bypasses expensive calls that require disk access. Configuring metadata caches also improves performance by reducing synchronization and spinlock contention when Adaptive Server has to retrieve index, object, or database information at run time.

For more information on configuring metadata caches, see "Metadata Caches" on page 11-61 in the *System Administration Guide*.

Monitor Access to SQL Batch Text

The monitor access to SQL batch text feature allows a user with the System Administrator role to display the SQL batch text of any client command that is running on Adaptive Server. Although you configure Adaptive Server to save the SQL batch text, the batches are viewable only through Adaptive Server Monitor™ Server.

Monitor access includes **sp_showplan**, which allows you to view the query plan for the query that is currently running. Monitor Server is not required to retrieve the query plans.

Both monitor access and **sp_showplan** are particularly useful for determining why a client session is hanging or using large amounts of CPU time or physical I/O.

Allocating Memory for Batch Text

Monitor access to SQL batch text is turned off by default when Adaptive Server is installed, so you must configure Adaptive Server to allocate memory for this feature. Consider the following to help you determine how much memory to allocate per user:

- SQL batches exceeding the allocated amount of memory are truncated without warning. Therefore, if you do not allocate

enough memory for the batch statements, the text you want to view might be the section of the batch that is truncated.

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

New Columns in *sysprocesses*

The procedure you are viewing may be nested in a batch of SQL text. To aid in the viewing of nested procedure, new columns have been added to the *sysprocesses* table to specify the object ID, current statement number and line number of the current statement of the procedure that is currently running.

Table 1-14 shows the new columns added to the *sysprocesses* table.

Table 1-14: New columns in sysprocesses

Column Name	Datatype	Specifies
id	*integer*	The object ID of the running procedure; if 0, no procedure is running.
linenum	*integer*	The line number of the current statement within the running stored procedure, or the line number of the current SQL batch statement, if no procedure is running.
stmtnum	*integer*	The current statement number within the running procedure, or the SQL batch statement number, if no procedure is running.

For more information, see “Configuring Adaptive Server to Save SQL Batch Text” on page 4-17 of the *System Administration Guide*.

Parallel Bulk Copy

Parallel bulk copy allows you to copy data in parallel to Adaptive Server from multiple files. Parallel bulk copy substantially increases performance during **bcp** sessions because large bulk copy jobs can be split into multiple sessions and run concurrently.

Before you copy data into your database using parallel bulk copy, you must first partition the table that will contain the data. You can then issue multiple **bcp** sessions, with each session copying its data from a different source file into a different partition. If you are copying data from one, large operating system file, you divide this file into smaller files and then start multiple parallel bulk copy sessions.

Figure 1-2 illustrates how parallel bulk copy works.

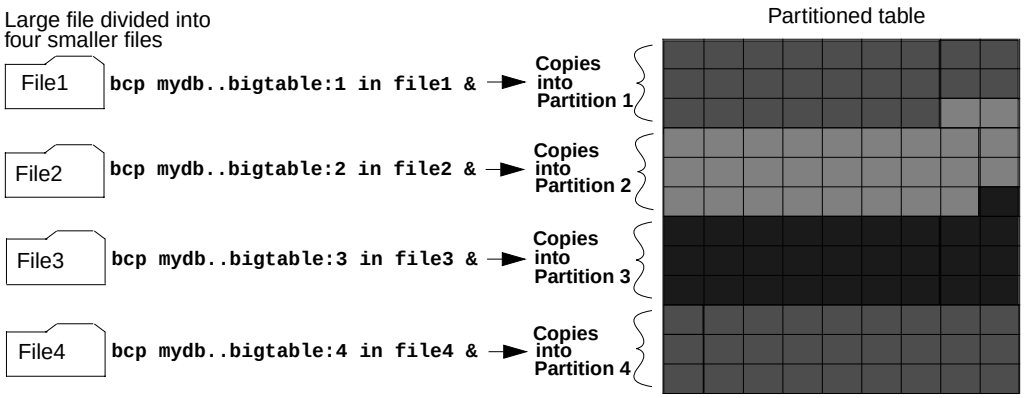

Figure 1-2: Copying data into a partitioned table using parallel bcp

You can also use the **-F** and **-L** flags to designate the first and last row of the same host file for each of the multiple parallel bulk copy sessions.

For more information about parallel bulk copy, see **bcp** in the *Utility Programs* manual for your platform.

Parallel Queries and Enhanced Partitioning

When you configure Adaptive Server for parallel query processing, the optimizer evaluates each query to determine whether it is eligible for parallel execution. If it is, the query is divided into components

that process simultaneously. The results are combined and delivered to the client in a shorter period of time than it would take to process the query serially as a single component.

For the same throughput, processing a query in parallel requires more work from Adaptive Server and additional resources than processing a query in serial. It also involves evaluating more complex trade-offs to achieve optimal performance. Fully enabled parallel query processing requires multiple processes, engines, and partitions, resulting in increased overhead for Adaptive Server, additional CPU requirements, and increased disk I/O.

You can configure various levels of parallelism, each providing a performance gain and requiring corresponding trade-offs in physical resources. Chapter 13, "Introduction to Parallel Query Processing," in the *Performance and Tuning Guide* introduces the Adaptive Server parallel query processing model and concepts. It also discusses the trade-offs between resources and performance gains for different levels of parallelism.

For More Information

In the *Performance and Tuning Guide*, see:

- Chapter 5, "Locking in Adaptive Server," for information on how Adaptive Server supports locking for parallel query execution.
- Chapter 9, "Understanding Query Plans," for information on new **showplan** messages added for parallel query execution.
- Chapter 14, "Parallel Query Optimization," for details on how the Adaptive Server optimizer determines eligibility for parallel execution.
- Chapter 15, "Parallel Sorting," to understand parallel sorting topics.
- Chapter 17, "Controlling Physical Data Placement," for information on partitioned tables, creating clustered indexes on partitioned tables, and parallel processing.

In the *System Administration Guide*, see:

- Chapter 11, "Setting Configuration Parameters," for details on how you can configure various levels of parallelism, each providing a performance gain and requiring corresponding trade-offs in physical resources.

Point-in-Time Recovery

Point-in-time recovery allows you to recover a database by rolling it forward to a particular time in its transaction log. You do this by using the **until_time** option of the **load transaction** command.

This feature is useful if, for example, a user inadvertently drops an important table from a database; you can use the **until_time** option to recover the changes to that database up to a time just before the table was dropped.

You can use point-in-time recovery for any database that can be loaded and dumped. However, you cannot use it for databases in which the data and logs are on the same device, such as the *master* database or for any database, such as *tempdb*, whose log has been truncated since the last **dump database**.

For more information on point-in-time recovery, see "Loading a Transaction Log to a Point in Time" on page 21-44 in the *System Administration Guide* and the **load transaction** command in the *Adaptive Server Reference Manual*.

Proxy Authorization

Adaptive Server provides the proxy authorization capability, which allows one user to assume the identity of another user on a server-wide basis. A System Security Officer can grant the ability to assume the security context of another user to selected logins. If a login has permission to use proxy authorization, the login can impersonate any other login in Adaptive Server.

◆ ***WARNING!***

The ability to assume another user's identity is extremely powerful and must be strictly limited. A user with this permission could assume the identity of the "sa" login, which would give the user unlimited power within Adaptive Server. Plan to limit this permission to trusted administrators and applications, and to audit their server activity.

A System Security Officer or a System Administrator might want to assume the permissions of another user to make sure the permissions are correct for a user or to perform maintenance on a user's database objects.

An application server can use proxy authorization to login to Adaptive Server with a generic login, which the application server uses to execute procedures and commands for several users.

A System Security Officer uses the **grant set proxy** or **grant set session authorization** command to give a user permission to use proxy authorization. The user with this permission can then execute either **set proxy** or **set session authorization** to become another user. A user executing **set proxy** or **set session authorization** operates with both the login and server user ID of the user being impersonated. The login and server user ID are active across the entire server in all databases.

➤ ***Note***

set proxy and **set session authorization** are identical in function and can be used interchangeably. The only difference between them is that **set session authorization** is SQL92 compatible, and **set proxy** is a Transact-SQL extension.

For more information about proxy authorization, see the *Security Administration Guide.*

Recovery Fault Isolation

Prior to this release, when recovery detected corruption in a database, it would make the entire database inaccessible. The database remained unavailable to users until the suspect pages had been repaired or removed from the database.

With recovery fault isolation, a System Administrator has the option of isolating the corrupt pages marked suspect by recovery while the rest of the database is brought online and made available to users.

Recovery fault isolation provides a System Administrator the ability to:

- Specify whether an entire database or just the suspect pages become inaccessible when recovery detects corruption
- Specify whether a database with suspect pages is brought online in **read_only** mode
- List databases that have suspect pages
- List suspect pages by page ID, index ID, and object name
- Bring suspect pages online for the System Administrator while they are being repaired

- Bring suspect pages online for all database users after they have been repaired

The ability to isolate the suspect pages while bringing the rest of the database online provides a greater degree of flexibility in dealing with data corruption. Problems can be diagnosed, and sometimes corrected, while most of the database is accessible to users. The extent of the damage can be assessed and emergency repairs can be scheduled for a convenient time.

A System Administrator can force offline pages online temporarily, for System Administrator use only, to aid in the analysis and repair of the suspect pages.

Recovery fault isolation applies only to user databases. System databases are always taken entirely offline if they have any suspect pages.

New System Procedures

Table 1-15 lists the system procedures that support recovery fault isolation:

Table 1-15: New system procedures for recovery fault isolation

System Procedure	Description
sp_forceonline_db	Provides access to all the suspect pages in a database.
sp_forceonline_page	Provides access to an individual suspect page.
sp_listsuspect_db	Lists all databases that have pages marked suspect by recovery.
sp_listsuspect_page	Lists all pages in a database that are marked suspect by recovery.
sp_setsuspect_granularity	Displays and sets the recovery fault isolation mode. This mode governs whether recovery marks an entire database or only the corrupt pages suspect.
sp_setsuspect_threshold	Sets the maximum number of suspect pages that recovery will allow in a database before marking the entire database suspect.

Side Effects of Recovery Fault Isolation

- Transactions that attempt to access offline data directly or indirectly will fail.
- You cannot use **dump database** on a database that has offline pages. This prevents users from dumping a corrupt database.
- You cannot use **dump transaction with no_log** or **dump transaction with truncate_only** on a database that has offline pages.

For more information on recovery fault isolation, see "Fault Isolation During Recovery" on page 20-7 in the *System Administration Guide*.

Relaxed LRU Cache Replacement Policy

Relaxed LRU (least recently used) cache replacement policy is a performance feature designed to reduce overhead in named caches where little or no buffer replacement is occurring.

Relaxed LRU replacement policy removes the cache linkage overhead required by caches maintained on an MRU / LRU (most recently used / least recently used) basis. Relaxed LRU replacement policy is meant to be used when both of the following conditions are true:

- There is little or no replacement of buffers in the cache, and
- The data is not updated or is updated rarely.

In addition to saving the overhead to maintain the MRU / LRU chain, relaxed LRU replacement policy can improve CPU performance on multiple-CPU servers.

See Chapter 9, "Configuring Data Caches," in the *System Administration Guide* for information on configuring caches to use LRU cache replacement policy.

See Chapter 16, "Memory Use and Performance," in the *Performance and Tuning Guide* for information on how to select caches where relaxed LRU replacement policy can improve performance.

Resource Limits

Adaptive Server provides resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. A **resource limit** is a set of parameters specified by

a System Administrator to prevent an individual login or application from:

- Exceeding estimated or actual I/O costs, as determined by the optimizer
- Returning more than a set number of rows
- Exceeding a given elapsed time

The set of parameters for a resource limit includes the time of day to enforce the limit and the type of action to take. For example, you can prevent huge reports from running during critical times of the day or kill a session whose query produces unwanted **Cartesian products**.

To specify times of the day or week, you can create named time ranges. A **time range** is a contiguous block of time within a single day across one or more contiguous days of the week. It is defined by its starting and ending periods. Contiguous days of the week are allowed to wrap around the end of one week to the beginning of the next.

Adaptive Server includes one predefined time range, the "at all times" range, which covers the period of midnight to midnight, Monday through Sunday. You can create, modify, and drop additional time ranges as necessary for resource limits.

For more information on resource limits, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide.*

New System Procedures

The system procedures in Table 1-16 support the resource limits feature.

Table 1-16: New system procedures for resource limits

System Procedure	Description
sp_add_resource_limit	Creates a resource limit.
sp_add_time_range	Creates a named time range.
sp_configure	

Table 1-16: New system procedures for resource limits

System Procedure	Description
"allow resource limits"	Signals the server to allocate internal memory for time ranges, resource limits, and internal server alarms. It also signals the server to internally assign applicable ranges and limits to user sessions. When this option is set to 1 (on), **sp_configure** displays the optimizer's cost estimate for a query. The default is 0 (off).
sp_drop_resource_limit	Drops a resource limit.
sp_drop_time_range	Drops a named time range.
sp_help_resource_limit	Displays resource limits in the current server.
sp_modify_resource_limit	Modifies a resource limit.
sp_modify_time_range	Modifies a named time range.

Supporting Commands

The following commands support the resource limits feature:

- **set showplan**
- **set statistics io**
- **statistics time**
- **select** *@@rowcount*

Source Text Verification and Encryption

In releases prior to Adaptive Server release 11.5, the **source text** of **compiled objects** was saved in *syscomments* only so that it could be returned to a user who executed **sp_helptext**. Since this was the only reason for saving the source text, users often deleted it to save disk space and to remove confidential information from this public area.

In Adaptive Server release 11.5, you should not remove the source text from *syscomments*. Deleting the source text from *syscomments* will cause problems when upgrading to the next release of Adaptive Server. The upgrade process in future releases of Adaptive Server will require that the source text of compiled objects be present in *syscomments* at the time of the upgrade. Before upgrading to the next release of Adaptive Server, you need to restore missing source text to *syscomments*.

You can use the **sp_checksource** system procedure to verify that source text is present in *syscomments* for each compiled object. To prevent users from seeing the source text in *syscomments*, you can encrypt the text with the **sp_hidetext** system procedure.

For more information, see "Compiled Objects" on page 1-3 in the *Transact-SQL User's Guide*. Also see **sp_checksource** and **sp_hidetext** in the *Adaptive Server Reference Manual*.

Sybase Central

With the release of Adaptive Server, Sybase no longer supplies SQL Server Manager, which has been replaced by Sybase Central™, Sybase's common management interface, and the Adaptive Server Enterprise plug-in for Sybase Central.

The Adaptive Server plug-in for Sybase Central allows you to manage Adaptive Server installations using the Sybase Central graphical management tool. You can manage Adaptive Server running on any platform from a PC that is running Windows 95 or Windows NT.

To learn how to use Sybase Central to manage Adaptive Server, see *Managing and Monitoring Sybase Adaptive Server Enterprise*.

sp_sysmon

sp_sysmon supports the new performance features in release 11.5, including the addition of these new sections in the **sp_sysmon** report:

- Application Queue Management
- Asynchronous Prefetch Usage
- Metadata Cache Management
- Monitor Access to Executing SQL
- Parallel Query Management
- Worker Process Management

The syntax for **sp_sysmon** has been changed:

- You can specify the interval using the form "hh:mm:ss".
- You can use the **begin_sample** argument to start **sp_sysmon** which continues to monitor until you use **end_sample** to print the report.

- You can print only one section of the report at a time. This example specifies an interval of 15 minutes and prints only the Data Cache Management section:

  ```
  sp_sysmon "00:15:00, dcache
  ```

See Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide* for more information.

Two-Phase Commit Enhancements

The two-phase commit protocol allows client applications to coordinate transaction updates across two or more Adaptive Servers and treats those transactions as a single transaction. Two-phase commit guarantees that either all or none of the databases in the participating Adaptive Servers are updated.

The enhancements to two-phase commit are:

- Improved space and log management – The *spt_committab* table has been moved from the *master* database to a new system database, *sybsystemdb*.
- Improved performance – You can use a random value for the commit identifier—*commid*.

spt_comittab Moved to a New System Database (*sybsystemdb)*

The *spt_committab* table in Adaptive Server stores information and tracks the completion status of each two-phase commit transaction. All activities made against *spt_committab* are logged. Prior to 11.5, *spt_committab* resided in the *master* database. However, the logging activity against *spt_committab* for two-phase commit transactions can fill up the master transaction log, causing frequent space management problems.

In release 11.5, the *spt_committab* table resides in a new system database called *sybsystemdb,* instead of the *master* database. Providing *spt_committab* its own database makes space management and recovery of two-phase commit transactions easier, for two reasons:

- The **dump tran** command can be used against *sybsystemdb*.
- Unlike the *master* database, the *sybsystemdb* database is extensible.

Commit Identifier Changed to a Random Value

The value of the commit identifier, *commid* in the *spt_committab* table, becomes a random value (between 1 and 2,147,483,647 (2^{31}-1), instead of a monotonically increasing value. This reduces page lock contention on the row where the maximum *commid* resides when several distributed transactions try to insert an entry into the *spt_committab* table. *spt_committab* continues to have a unique, clustered index.

For information on how to configure Adaptive Server for two-phase commit functionality, see the Adaptive Server configuration documentation for your platform.

User-Defined Roles

The user-defined roles feature allows a System Security Officer to simplify the task of granting and preventing access to data.

The System Security Officer can create roles, request object owners to grant privileges to each role, and grant the user-defined roles to individual employees, based on the functions they perform in the organization. The System Security Officer can also revoke user-defined roles granted to the employee.

For more information on user-defined roles, see the *Security Administration Guide*.

Windows NT Event Log Integration

This release has expanded support for logging Adaptive Server errors in the Windows NT Event Log. This is in addition to automatic logging in the Adaptive Server error log. This option is available on Windows NT servers only.

By default, Windows NT Event Logging of Adaptive Server messages is enabled, but you can disable it by setting the **event logging** configuration parameter to 0. You can also specify a remote Windows NT computer to receive event logging by setting the **event log computer name** configuration parameter to the name of the remote computer.

This feature provides an extended stored procedure that you can use to log user-defined events to the Windows NT Event Log from within Adaptive Server. See **sp_logdevice** in the *Adaptive Server Reference Manual*.

Supporting System ESPs and System Procedures

The following system ESP and **sp_configure** parameters support the Windows NT Event Log Integration feature.

Table 1-17: System support for Windows NT Event Log Integration

Procedure	Description
xp_logevent	A system-defined ESP that provides for logging a user-defined event in the Windows NT Event Log.
sp_configure	
"event log computer name"	Specifies the name of the Windows NT PC that logs Adaptive Server messages in its Windows NT Event Log.
"event logging"	Enables and disables the logging ofAdaptive Server messages in the Windows NT Event Log.

Windows NT MAPI Support

Sybase Adaptive Server Enterprise for Windows NT can send and receive email messages using Microsoft's Message Application Programming Interface (MAPI). This facility, called Sybmail, is currently available only on Windows NT.

Adaptive Server can send outgoing messages that consist of text or query results. Adaptive Server can receive queries as incoming messages.

To use Sybmail, Adaptive Server must have a Windows NT Mail account and a Windows NT Mail profile. Sybmail must also have a login in Adaptive Server.

See *Configuring Adaptive Server for Windows NT* for information on setting up and using Sybmail.

Supporting System ESPs and System Procedures

The following system ESPs and system procedures support the Windows NT MAPI feature:

Table 1-18: System support for Windows NT MAPI

Procedure	Description
sp_configure	
"start mail session"	Enables and disables the automatic initiation of a Adaptive Server mail session when Adaptive Server is started.
sp_processmail	Reads, sends, and deletes messages in the Adaptive Server message inbox. Available on Windows NT only.
xp_deletemail	Deletes a message from the Adaptive Server message inbox.
xp_findnextmsg	Retrieves the message identifier of the next message in the Adaptive Server message inbox.
xp_readmail	Reads a message from the Adaptive Server message inbox.
xp_sendmail	Sends a message to the specified recipients using the MAPI interface.
xp_startmail	Starts an Adaptive Server mail session.
xp_stopmail	Stops an Adaptive Server mail session.

For more information on the system-defined ESPs, see Chapter 5, "System Extended Stored Procedures," in the *Adaptive Server Reference Manual*.

Windows NT Performance Monitor Integration

You can enable and disable the ability to monitor Adaptive Server statistics from the Windows NT Performance Monitor by setting the **SQL Perfmon Integration** configuration parameter. This parameter applies to Windows NT servers only.

Adaptive Server must be registered as an NT Service to support performance monitor integration. This occurs automatically when you start Adaptive Server using Sybase Central, the Services Manager, or the Services option in the Control Panel or when you

have configured Windows NT to start Adaptive Server as an automatic service.

For more information on Windows NT Performance Monitor, see Chapter 9, "Administrative Tasks and Performance and Tuning," in *Configuring Adaptive Server for Windows NT*.

Supporting System Procedure

The following **sp_configure** configuration parameter supports the Windows NT Performance Monitor Integration feature:

Table 1-19: System support for Windows NT Performance Monitor

Procedure	Description
sp_configure	
"SQL Perfmon Integration"	Enables and disables the ability to monitor Adaptive Server statistics from the Windows NT Performance Monitor.

2 System Changes in Adaptive Server Release 11.5

This chapter describes the system changes introduced in Adaptive Server release 11.5.

Topics covered are:

System Changes in Release 11.5

This section provides a summary of the release 11.5 changes, as follows:

New Installation and Upgrade Utilities for UNIX

For Adaptive Server release 11.5, **sybinit**, the command line installation program on UNIX platforms, has been replaced by a set of easy-to-use graphical installation and upgrade utilities. These utilities are accessible from a common desktop application, **sybsetup**, that you can download from your Sybase CD. You need an X Windows / Motif-based system interface to use the new utilities in graphical mode. You can also execute the utilities in UNIX command mode.

The new utilities are:

- **srvbuild** – creates a new Adaptive Server, Backup Server™, Monitor Server, or XP Server with default or user-specified values for key configuration attributes.
- **sqlupgrade** – upgrades Adaptive Server from a previous release.
- **sqlloc** – installs and modifies languages, character sets, and sort order defaults for Adaptive Server.

The following installation utility is based on **sybinit** and operates only in UNIX command mode.

- **auditinit** – installs and configures C2 auditing in Adaptive Server.

Table 2-1 lists the documents that describe the new utilities.

Table 2-1: Additional information on installation and upgrade utilities

Utility	See
auditinit	• *Configuring Adaptive Server for UNIX Platforms*
sqlloc	• *Installing Adaptive Server and OmniConnect on UNIX Platforms* • *Configuring Adaptive Server for UNIX Platforms* • Online help • *Utility Programs for UNIX Platforms* (to use in UNIX command mode)
sqlupgrade, **srvbuild**, or **sybsetup**	• *Installing Adaptive Server and OmniConnect on UNIX Platforms* • Online help • *Utility Programs for UNIX Platforms* (to use in UNIX command mode)

New Utilities for Editing the Interfaces File

Release 11.5 provides new utilities for editing the interfaces file on UNIX and Windows NT platforms.

For UNIX

In previous releases, **sybinit** was used to edit the interfaces (*interfaces*) file on UNIX platforms. For Adaptive Server release 11.5, **sybinit** has been replaced by **dsedit** and **dscp**. You need an X Windows / Motif-

based system interface to use **dsedit**. **dscp** operates only in UNIX command-line mode.

For more information on **dsedit** and **dscp**, see the following:

- *Installing Adaptive Server and OmniConnect on UNIX Platforms*
- *Configuring Adaptive Server for UNIX Platforms*
- *Utility Programs for UNIX Platforms*

For Windows NT

In previous releases, **sqledit** was used to edit the interfaces file (*sql.ini*) on Windows NT platforms. For Adaptive Server release 11.5, **sqledit** has been replaced by **dsedit**.

For more information on **dsedit** on Windows NT, see the following:

- *Installing Adaptive Server and OmniConnect on Windows NT*
- *Configuring Adaptive Server for Windows NT*
- *Utility Programs for Windows and Windows NT*

Differences Between CIS and OmniConnect

Users of OmniConnect release 10.5, who are upgrading to Adaptive Server release 11.5, should note the system differences described in Table 2-2:

Table 2-2: Differences between CIS and OmniConnect

Area Affected	Description of Change
C-ISAM	Support for C-ISAM has been dropped in release 11.5.
dbcc trace flags	See the *Component Integration Services User's Guide*.
Error messages	See the *Error Messages* manual.
Installation	CIS is installed using the **sybsetup** utility. See the installation documentation for your platform.
Server classes	A new server class called *sds* has been added to CIS. See the *Component Integration Services User's Guide* for more information.
System tables	The system tables used by CIS are different from those in OmniConnect 10.5. See the *Adaptive Server Reference Manual* for more information.

Table 2-2: Differences between CIS and OmniConnect (continued)

Area Affected	Description of Change
Utilities	The **defgen** and **ddlgen** utilities have been enhanced in CIS. See the *Component Integration Services User's Guide* for complete information.

Changes to Configuration Parameters

This section describes the new and changed configuration parameters in release 11.5.

New Configuration Parameters

Table 2-3 summarizes the new configuration parameters:

Table 2-3: New configuration parameters

Parameter	Function
allow backward scans	Enables or disables a performance enhancement that allows scanning a table or index in reverse page-chain order by a **select** query with the **desc** keyword. The default is 1, backward scans allowed.
allow resource limits	Signals the server to allocate internal memory for time ranges, resource limits, and internal server alarms. Also signals the server to internally assign applicable ranges and limits to user sessions. When this parameter is set to 1 (on), **sp_configure** displays the optimizer's cost estimate for a query. The default is 0 (off).
auditing	Enables or disables auditing for Adaptive Server. The parameter takes effect immediately upon execution. Auditing occurs only when this parameter is on.
audit queue size	Establishes the size of the audit queue. Because this parameter affects memory allocation, it does not take effect until Adaptive Server is restarted.
cis bulk insert batch size	Specifies how many rows are to be inserted before a bulk commit operation is invoked using Component Integration Services.

Table 2-3: New configuration parameters (continued)

Parameter	Function
cis connect timeout	Specifies the wait time for a Client-Library™ connection to succeed when using Component Integration Services.
cis cursor rows	Specifies a cursor row count, which takes effect during cursor open and cursor fetch operations.
cis packet size	Specifies a packet size for connections to remote servers via Client-Library.
cis rpc handling	Specifies the default method for RPC handling. If it is set to 0, the default, Component Integration Services uses the Adaptive Server site handler as the default RPC handling mechanism. If it is set to 1, Component Integration Services uses its own the access methods.
current audit table	Sets the current audit table. This takes effect immediately upon execution.
disable character set conversion	Disables character set conversion on data moving between clients and Adaptive Server. The default is 0 (enabled); a value of 1 turns off character set conversion (disabled).
enable cis	Determines whether Component Integration Services is enabled.
enable rep agent threads	For users of Replication Server®, the **enable rep agent theads** parameter enables the RepAgent thread within Adaptive Server.
esp execution priority	Sets the priority of the XP Server thread for ESP execution.
esp execution stacksize	Sets the size of the stack, in bytes, to allocate for ESP execution.
esp_unload_dll	Specifies whether DLLs that support ESPs should automatically be unloaded from XP Server memory after the ESP call has completed.
event log computer name	Specifies the name of the Windows NT PC that logs Adaptive Server messages in its Windows NT Event Log.
event logging	Enables and disables the logging of Adaptive Server messages in the Windows NT Event Log.
global async prefetch limit	Specifies the default asynchronous prefetch limit for all pools not explicitly configured with **sp_poolconfig**.

Table 2-3: New configuration parameters (continued)

Parameter	Function
log audit logon failure	Specifies whether to log unsuccessful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.
log audit logon success	Specifies whether to log successful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.
max cis remote connections	Specifies the maximum number of concurrent Client-Library connections that Component Integration Services can make to remote servers.
max cis remote servers	Specifies the number of SRVDES data structures that should be allocated by the server during start-up.
max parallel degree	Specifies the server-wide maximum degree of parallelism (the maximum number of worker processes per table or index scan) for any one query.
max scan parallel degree	Specifies the maximum server-wide degree of parallelism for parallel index scans for partitioned tables and nonpartitioned tables, and for parallel table scans for nonpartitioned tables.
max SQL text monitored	Specifies the amount of memory allocated per user connection for saving SQL batch text.
memory per worker process	Specifies the amount of memory allocated for use by worker processes.
msg confidentiality reqd	Requires all messages into and out of Adaptive Server, to be encrypted. If this parameter is 0 (the default), message confidentiality is not required, but may be established by the client.
msg integrity reqd	Specifies that all messages be checked for tampering. If this parameter is 0 (the default), message integrity is not required, but may be established by the client.
msg origin checks reqd	Specifies that all messages be checked for tampering. If this parameter is 0 (the default), message integrity is not required, but may be established by the client.

Table 2-3: New configuration parameters (continued)

Parameter	Function
msg out-of-seq checks reqd	Specifies that all messages must be checked to make sure that the sequence has not changed. If this parameter is 0 (the default), sequence checking is not required, but may be established by the client.
msg replay detection reqd	Requires that all messages be checked to ensure that they have not been replayed or intercepted. If this parameter is 0 (the default), replay detection is not required, but may be established by the client.
number of aux scan descriptors	Sets the number of auxiliary scan descriptors needed for a server.
number of large i/o buffers	Sets the number of 16K buffers reserved for performing large I/O tasks, such as **load database**. The default is 6.
number of open indexes	Sets the maximum number of indexes that can be used simultaneously on Adaptive Server; also sets the number of metadata index descriptors used in the server.
number of worker processes	Specifies the maximum number of worker processes that can be in use in the system at any one time.
open index hash spinlock ratio	Sets the number of index metadata descriptor hash tables protected by one spinlock.
open index spinlock ratio	Specifies the number of index metadata descriptors protected by one spinlock.
open object spinlock ratio	Specifies the number of object metadata descriptors protected by one spinlock.
secure default login	Establishes logins for users who are already authenticated by a security mechanism, but who do not have their own login names in *master..syslogins*.
SQL Perfmon Integration	Enables and disables the ability to monitor Adaptive Server statistics from the Windows NT Performance Monitor.
start mail session	Enables and disables the automatic initiation of an Adaptive Server mail session when Adaptive Server is started.

Table 2-3: New configuration parameters (continued)

Parameter	Function
suspend auditing when full	Controls the behavior of the audit process when an audit device becomes full. This takes effect immediately upon execution.
unified login	Specifies that all users must be authenticated by a security mechanism before logging into Adaptive Server. A value of 1 requires unified login; a value of 0 causes Adaptive Server to accept traditional login names and passwords, as well as already authenticated certificates.
use security services	Enables network-based security. If this parameter is 0 (the default), network-based security services such as unified login and message confidentiality are not available. This parameter does not take effect until Adaptive Server is restarted.
xp_cmdshell_context	Sets the security context for the operating system command to be executed using the **xp_cmdshell** system ESP.

Changes to Configuration Parameters

Table 2-4 summarizes the changes to existing configuration parameters.

Table 2-4: Changed configuration parameters

Parameter	Change
max number network listeners	The default value has been reduced from 15 to 5.
number of locks	The definition has been changed to include discussion of setting the number of locks needed for parallel execution.
number of open databases	Now sets the number of metadata database descriptors used for the server.
number of open objects	Now sets the number of metadata object descriptors used for the server.

Obsolete Configuration Parameters

The **sp_configure** parameters in the following table are no longer valid with this release:

Table 2-5: Obsolete configuration parameters

Parameter	Description
number of extent i/o buffers	It is no longer necessary to reserve dedicated sort buffers with the **number of extent i/o buffers** for large I/Os. Instead, sort operations can use buffer pools configured for large I/Os from default and named data caches.
sort page count	**number of sort buffers** is the only user-configurable parameter for sorting.

Prior to Adaptive Server release 11.5, the sort manager issued its own I/O requests without using the buffer pools in data caches. To facilitate large I/Os, the System Administrator could reserve dedicated sort buffers with the **number of extent i/o buffers** configuration parameter. In release 11.5, all sort operations can benefit from the features of the buffer pools within data caches. For more information, see Chapter 9, "Configuring Data Caches," in the *System Administration Guide*.

Changes to the Server Configuration File

For installations of release, the default configuration file is used.

For upgrades to release 11.5 , using a server's existing configuration file, new parameters are not included in the configuration file. No errors are reported for the missing rows when Adaptive Server is started, and all new configuration parameters are set to their default values. The first time an **sp_configure** or **sp_cacheconfig** command writes the configuration file, rows for the new values are included in the file.

In release 11.5, some configuration parameters have been removed. If you start Adaptive Server with a configuration file that contains these options, error messages are printed in the error log. The next time the configuration file is written by an **sp_configure** or **sp_cacheconfig** command, the obsolete parameters are not included.

Changes for Asynchronous Prefetch

The default for the global asynchronous prefetch limit is 10 percent.

The following portion of a configuration file shows **global asynch prefetch limit** with its default value:

```
[Cache Manager]
        number of oam trips = DEFAULT
        number of index trips = DEFAULT
        procedure cache percent = DEFAULT
        memory alignment boundary = DEFAULT
        global async prefetch limit = DEFAULT
```

If you want to disable asynchronous prefetch before starting Adaptive Server, you can edit the "global async" line (if it exists) or add it to an existing configuration file, in the position shown in the preceding example. The syntax to disable asynchronous prefetch is:

```
global async prefetch limit = 0
```

unique auto_identity index Database Option

The **sp_dboption** system procedure has a new option, **unique auto_identity index**. When the **unique auto_identity index** option is set to **true**, it adds an IDENTITY column with a unique, nonclustered index to each new table. This option allows you to create tables that use IDENTITY columns and are used with updatable cursors. A table being used in an updatable cursor must have a unique index to ensure that the cursor will be positioned at the correct row after a **fetch**.

For more information, see **unique auto_identity index** in Chapter 16, "Setting Database Options," in the *System Administration Guide*.

Setting Local Variables in *update* Statements

The **update** command has been enhanced so that you can set values for variables in an **update** statement similar to the way you can set them in a **select** statement. This provides an OLTP performance enhancement that reduces lock contention and CPU consumption associated with the extra **select** statements required in pre-11.5 **update** statements.

Changes to Transact-SQL Commands

This section describes the new and changed Transact-SQL commands in release 11.5.

New Transact-SQL Commands

Table 2-6 summarizes the new Transact-SQL commands.

Table 2-6: New commands

Command	Function
case	Indicates the beginning of a **case** expression.
coalesce	Searches for the first occurrence of a non-NULL value.
nullif	Compares the values of *value1* and *value2*. If *value1* equals *value2*, **nullif** returns NULL; if *value1* does not equal *value2*, **nullif** returns *value1*.
update all statistics	Updates all statistics information for a given table.
update partition statistics	Updates information about the number of pages in each partition for a partitioned table.
when...then	Searches for occurrences of *search_condition1* and replaces it with *result1*. This value becomes the value of **case**.

Changed Transact-SQL Commands

Table 2-7 summarizes the changes to existing Transact-SQL commands:

Table 2-7: Changed commands

Command	Change
create procedure	Includes syntax for creating an extended stored procedure (ESP).
drop procedure	Drops an ESP (as well as a stored procedure).
execute	Executes an ESP (as well as a stored procedure).
load transaction	Includes the **until_time** option, which allows you to recover a database to a specified time.
select	Includes the **parallel** option to limit the number of worker processes used in a query.
set	For new and changed options, see Table 2-8.

New and Changed *set* Command Options

The following table summarizes the new and changed options for the **set** command:

Table 2-8: New and changed set command options

set Option	Description
parallel degree	Specifies an upper limit for the number of worker processes used in the parallel execution of a query.
process_limit_action	Specifies whether Adaptive Server executes parallel queries when an insufficient number of worker processes are available.
scan_parallel_degree	Reduces the session-specific degree of parallelism for parallel index scans and parallel table scans on nonpartitioned tables.
showplan	Messages added for parallel processing.
statistics io	Displays the number of logical and physical reads performed by asynchronous prefetch and the number of buffers fetched by asynchronous prefetch that were used by the query. If you have configured Adaptive Server to enforce resource limits, this option displays the total I/O cost.
statistics time	Displays the amount of time Adaptive Server uses to parse and compile each command.

New Transact-SQL Functions

Table 2-9 summarizes the new Transact-SQL functions.

Table 2-9: New Transact-SQL functions

Function	Description
is_sec_service_on	Determines whether a particular network-based security service is active.
ptn_data_pgs	Returns the number of data pages used by a partition.
show_sec_services	Lists the network-based security services that are active for the session.

New and Changed System Procedures

This section describes the new system procedures added and changes made to existing system procedures.

New System Procedures

Table 2-10 summarizes the new system procedures:

Table 2-10: New system procedures

System Procedure	Function
sp_addauditrecord	Adds user-defined audit records (comments) to the audit trail. Users can add these records only if a System Security Officer enables ad hoc auditing with **sp_audit**.
sp_addengine	Adds an engine to an existing engine group or, if the group does not exist, creates an engine group and adds the engine.
sp_addexeclass	Creates a user-defined execution class that you can bind to client applications, logins, and stored procedures. If the class already exists, the class attribute values are updated with the supplied values.
sp_addextendedproc	Creates an extended stored procedure (ESP).
sp_add_resource_limit	Creates a resource limit.
sp_add_time_range	Creates a named time range.
sp_altermessage	Enables and disables the logging of a message in the Adaptive Server error log and in the Windows NT Event Log, if event logging is enabled.
sp_audit	Enables and disables all auditing options. This is the only system procedure required to establish the events to be audited.
sp_bindexeclass	Associates an execution class with a client application, login, or stored procedure.
sp_clearpsexe	Clears the execution attributes of the client application, login, or stored procedure that was set by **sp_setpsexe**.
sp_countmetadata	Displays the number of indexes, objects, or databases (including system databases) in a server.
sp_displayaudit	Displays the active auditing options.

Table 2-10: New system procedures (continued)

System Procedure	Function
sp_dropengine	Drops an engine from a specified engine group. If the engine is the last one in the group, Adaptive Server also drops the group.
sp_dropexeclass	Drops a user-defined execution class.
sp_dropextendedproc	Drops an ESP.
sp_drop_resource_limit	Drops a resource limit.
sp_drop_time_range	Drops a named time range.
sp_familylock	Reports information about all locks held by a family (coordinating process and its worker processes) executing a statement in parallel.
sp_forceonline_db	Provides access to all suspect pages in a database.
sp_forceonline_page	Provides access to an individual suspect page.
sp_freedll	Unloads a DLL that was loaded into XP Server memory to support the execution of an ESP.
sp_helpconfig	Reports help information on configuration parameters. Also reports how much memory Adaptive Server needs when a configuration parameter is set to a specified size.
sp_helpextendedproc	Displays ESPs in the current database with their associated DLL files.
sp_help_resource_limit	Displays resource limits in the current server.
sp_listsuspect_db	Lists all databases that have pages marked suspect by recovery.
sp_listsuspect_page	Lists all pages in a database that are marked suspect by recovery.
sp_modify_time_range	Modifies a named time range.
sp_modify_resource_limit	Modifies a resource limit.
sp_monitorconfig	Displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases, such as the number of metadata descriptors currently used by the server. Also reports the number of auxiliary scan descriptors in use. A scan descriptor manages a single scan of a table when queries are run on that table.
sp_processmail	Reads, sends, and deletes messages in the Adaptive Server message inbox. Available on Windows NT only.

Table 2-10: New system procedures (continued)

System Procedure	Function
sp_setsuspect_granularity	Displays and sets the recovery fault isolation mode. This mode governs whether recovery marks an entire database or only the corrupt pages suspect.
sp_setsuspect_threshold	Sets the maximum number of suspect pages that recovery will allow in a database before marking the entire database suspect.
sp_setpsexe	Sets custom execution attributes "on the fly" for an active client application, login, or stored procedure.
sp_showcontrolinfo	Displays information about all engine group assignments and all bound client applications, logins, and stored procedures. Also provides information on an individual basis about application, login, or stored procedure bindings to an execution class; engine group compositions; and session-level attribute bindings.
sp_showexeclass	Displays the execution class attributes and the engines in any engine group associated with the specified execution class.
sp_showplan	Displays the query plan for any user connection for the current SQL statement (or a previous statement in the same batch). The query plan is displayed in **showplan** format.
sp_showpsexe	Displays execution class, current priority, and affinity for client application, login, and stored procedure processes running on Adaptive Server.
sp_unbindexeclass	Removes the execution class attribute previously associated with a client application, login, or stored procedure for the specified scope.

Changed System Procedures

Table 2-11 summarizes the changes made to existing system procedures.

Table 2-11: Changed system procedures

Procedure	Change
sp_addmessage	Adds parameters for specifying whether a message is always logged (**with_log**) and for overwriting an existing message of the same message number and language ID (**replace**).

Table 2-11: Changed system procedures (continued)

Procedure	Change
sp_cacheconfig	Adds a parameter to configure relaxed LRU cache policy. Output displays cache policy and asynchronous prefetch limit.
sp_configure	Adds options. See "New Configuration Parameters" on page 2-4.
sp_dboption	Has a new option, **unique auto_identity index**, that automatically adds an IDENTITY column with a unique, nonclustered index to new tables.
sp_drop_resource_limit	Drops all resource limits associated with the specified login.
sp_helpconstraint	In addition to reporting information about integrity constraints, **sp_helpconstraint** displays more details about the number of references used by a specified table.
sp_helpartition	Lists the number of data pages for each partition and provides summary information for the minimum, maximum, and average number of pages per partition.
sp_helpsegment	Prints the total number of pages, free pages, and used pages.
sp_lock	Adds a column called *fid* to identify the family associated with the task for parallel execution.
sp_poolconfig	Adds syntax to set the asynchronous prefetch limit for a pool.
sp_serveroption	Adds these options for network-based security: • **rpc security model A** • **rpc security model B** • **security mechanism** • **mutual authentication** • **use message confidentiality** • **use message integrity**
sp_who	Adds a column called *fid* to identify the family associated with the task for parallel execution.

Obsolete System Procedures

The system procedure in Table 2-12 is no longer valid with this release.

Table 2-12: Obsolete system procedure

System Procedure	Description
sp_auditoption	Use **sp_audit** for setting the options that you set with **sp_auditoption** in releases prior to 11.5.

Changes to Databases and System Tables

This section describes the changes made to databases and system tables to support release 11.5 features.

New Databases

Table 2-13 lists the databases that are new in this release:

Table 2-13: New databases

Database	Function
dbccalt	Used with **dbcc** commands to test severe cases of database corruption.
dbccdb	Stores the results of **dbcc** operations.
pubs3	Provides an updated version of the *pubs2* database.

New System Tables

Table 2-14 lists the system tables that are new in this release:

Table 2-14: New system tables

Table	Function
sysresourcelimits	Contains a row for each resource limit defined by Adaptive Server.

Table 2-14: New system tables (continued)

Table	Function
syssecmechs	Contains information about the security services supported by each security mechanism that is available to Adaptive Server. This is not a normal table. Rather, it is built dynamically when queried by a user.
systimeranges	Contains a row for each named time range defined by Adaptive Server to control when a resource limit is active.

Changed System Tables

Table 2-15 lists the system tables that have been changed in this release:

Table 2-15: Changed system tables

Table	Change
syscolumns	Adds the *remote_name* and *remote_type* columns.
syscomments	Adds the *status* column.
sysconfigures	Adds the *value4* column.
syscurconfigs	Adds the *apf_percent* column.
sysdatabases	Adds the *def_remote_loc* and *def_remote_type* columns.
sysindexes	Adds the *base_partition* column.
syslocks	Adds the *fid* and *context* columns.
syslogins	Adds the *srvname* column.
sysobjects	Adds the *versionts* column.
sysprocedures	Adds the *version* column.
sysprocesses	Adds the *fid*, *execlass*, *priority*, *affinity*, *id*, *stmtnum*, *linenum*, and *origsuid* columns.
sysservers	Adds the *srvclass* and *srvsecmech* columns.
syssrvroles	Adds the *password* column.
sysusermessages	Adds the *dlevel* column.

For more information, see Chapter 8, “System Tables,” in the *Adaptive Server Reference Manual*.

New Reserved Words

➤ ***Note***

If you are upgrading from 4.9.x to 11.5, be sure to check the reserved words created in releases 10.0 and 11.0 before upgrading to 11.5.
If you are upgrading from 10.0 to 11.5, be sure to check the reserved words created in release 11.0 before upgrading to 11.5.

The following keywords are new reserved words in this release:

- **activation**
- **connect**
- **consumers**
- **exclusive**
- **external**
- **identity_start**
- **membership**
- **passwd**
- **proxy**
- **session**

Reserved words cannot be used as object names or column names.

You must change all database names that are new reserved words before you can upgrade from an earlier release of the server. You can change table, view, and column names or use delimited identifiers. Once you upgrade to release 11.5, you cannot use database objects whose names are new reserved words until you modify your procedures, SQL scripts, and applications.

Changes to the Documentation

The Adaptive Server documentation set has been reorganized for this release as follows:

System Administration Guide

The *System Administration Guide* contains two new chapters.

- Chapter 12, “Limiting Access to Server Resources,” explains how to prevent queries and transactions from monopolizing server resources.
- Chapter 19, “dbccdb Tables,” describes the tables contained in the new *dbccdb* database.

Reference Supplement

The *Reference Supplement* no longer exists.

- The description of the *pubs2* database is now located in Appendix A, “The pubs2 Database,” in the *Transact-SQL User’s Guide.*
- The system table descriptions and reserved word lists are located in Chapter 8, “System Tables,” and Appendix B, “Reserved Words,” respectively, in the *Adaptive Server Reference Manual.*
- Error messages are now listed in the *Error Messages* manual.

Performance and Tuning Guide

The *Performance and Tuning Guide* contains three new chapters.

- Chapter 13, “Introduction to Parallel Query Processing,” introduces parallel processing on Adaptive Server.
- Chapter 21, “How Adaptive Server Uses Engines and CPUs,” gives an overview of processing and scheduling.
- Chapter 22, “Distributing Engine Resources Between Tasks,” describes the new engine affinity and execution precedence features.

Adaptive Server Reference Manual

The *Adaptive Server Reference Manual* includes two new chapters:

- Chapter 5, “System Extended Stored Procedures”
- Chapter 6, “dbcc Stored Procedures”

The *Reference Manual* no longer includes the Roadmap or the Topics chapter.

- The Roadmap, now called *Navigating the Documentation for Adaptive Server,* is available as a separate, online document. This online document provides a task-oriented approach to using Adaptive Server and provides links to the concepts and syntax that are relevant to each task.

- The descriptions of the Transact-SQL functions are now described in Chapter 2, "Transact-SQL Functions," in the *Transact-SQL User's Guide.*
- The information formerly in the Topics chapter is now in Appendix A, "Expressions, Identifiers, and Wildcard Characters," in the *Transact-SQL User's Guide*, the *System Administration Guide*, and the *Security Administration Guide.*

Transact-SQL User's Guide

The *Transact-SQL User's Guide* contains one new chapter and two new appendixes.

- Chapter 15, "Using Extended Stored Procedures"
- Appendix A, "The pubs2 Database"
- Appendix B, "The pubs3 Database"

In addition, information formerly in the "Transact-SQL Topics" section in the *Reference Manual* has been incorporated throughout the *Transact-SQL User's Guide*.

Adaptive Server Glossary

- The *Adaptive Server Glossary* is a new book that defines the glossary terms for most of the books in Adaptive Server documentation. Most books no longer contain individual glossaries.

Changes That May Affect Existing Applications

This section describes system changes introduced by release 11.5 that may affect your applications if you are upgrading to release 11.5 from an 11.0 release. Topics covered are:

Auditing Changes

Adaptive Server release 11.5 introduces a new auditing system. See "Auditing Enhancements" on page 1-3.

Beginning with this release, Adaptive Server no longer supports **sp_auditoption**, **sp_auditlogin**, **sp_auditdatabase**, **sp_auditobject**, and **sp_auditsproc**. The functionality provided by those system procedures is provided by the new system procedure **sp_audit**. The new auditing system continues to allow users to add their own audit records with **sp_addauditrecord**.

New Transact-SQL Keywords for 11.5

See "New Reserved Words" on page 2-19.

Order of Results and Parallel Processing

When Adaptive Server is configured for parallel query processing, parallel execution that uses continuous merge for an unsorted query returns results in a different order from a serial execution of the same query. This is because worker processes access and return different parts of the data simultaneously.

You may need to add an **order by** clause to queries to get consistently ordered results when you configure Adaptive Server for parallel processing.

Obsolete Configuration Parameters

See "Obsolete Configuration Parameters" on page 2-9.

Changes Due to Two-Phase Commit Enhancements

If you upgrade your existing two-phase commit configuration to the new 11.5 configuration, be aware of the following:

- User-written backup and recovery procedures for two-phase commit transactions should point to *sybsystemdb* instead of to *master*.
- Any user application or stored procedure should reference *sybsystemdb..spt_committab* instead of *master..spt_committab*.

Changes to *showplan* Output in Release 11.5

The output for **showplan** has been changed to include messages for parallel query optimization and referential integrity auxiliary scan descriptors. If any of your applications use **showplan** output, they may need to be changed to use the new output for parallel queries and scan descriptors.

See Chapter 9, "Understanding Query Plans," in the *Performance and Tuning Guide*.

Changes to System Procedure Output

If your applications use system procedure output, you may need to change them to accommodate the new output for changed system procedures.

The changes to system procedure output are as follows:

- The output for the system procedure **sp_helpartition** now contains a column containing the number of data pages for each partition in a partitioned table.
- The output for the system procedure **sp_helpconstraint** now displays more integrity constraint information than it did in release 11.0 or 11.0.x.
- The output for the system procedure **sp_helpserver** includes information about the security settings for remote procedures.

- The output for the system procedure **sp_lock** now displays a *fid* column to show the family ID for queries executed in parallel.
- The output for the system procedure **sp_who** now displays a *fid* column to show the family ID for queries executed in parallel.

See "Changed System Procedures" in Chapter 2, "System Changes in Adaptive Server Release 11.5" of this guide.

Increased Limit for Referential Integrity Queries

The maximum number of references allowed for a table has been increased from 16 to 192. If your applications use triggers as a workaround to accommodate the earlier 16-table limit for referential integrity queries, you may want to remove such triggers and redesign your tables to use declarative referential integrity. To check the declarative referential integrity used by your tables, run **sp_helpconstraint**, as described in the *Adaptive Server Reference Manual*.

For information on redesigning tables, see Chapter 7, "Creating Databases and Tables," in the *Transact-SQL User's Guide*.

Query Ordering Results and Component Integration Services

If a query using the **order by** clause is passed through a remote server through Component Integration Services, it may have different ordering results than if the query were run on a local server. If this affects your applications, you can disable this type of optimization by using **dbcc traceon(11216)** to set Component Integration Services to use pre-release 11.5 query processing methods.

Changed Behavior of the *datalength* Function

The behavior of the **datalength** function has been changed so that it always returns the exact length of a *char* value. In previous releases, SQL Server sometimes returned an incorrect value when certain string functions were used in conjunction with the **datalength** function on a *char* value.

The following example shows the use of the **rtrim** function to remove the trailing blank from a variable.

```
declare @v char(2)
select @v = rtrim(convert(char(2), 'a '))
select @v, datalength(@v),
   datalength(rtrim(convert(char(2), 'a ')))
```

The example output below illustrates how running the above **datalength** statements returns results that differ from the results in previous releases of SQL Server.

Pre-11.5 releases of SQL Server returned a data length of 1 for the *@v* variable:

```
(1 row affected)
-- ----------- -----------
a             1           1
(1 row affected)
```

Adaptive Server returns a data length of 2 for the *@v* variable:

```
(1 row affected)
-- ----------- -----------
a             2           1
(1 row affected)
```

To keep the behavior of existing applications consistent when you run them in Adaptive Server, change the variable declarations from *char* to *varchar*. For example, change the following declaration:

```
declare @month char(2)
```

to:

```
declare @month @varchar(2)
```

Trailing Spaces of *char* Data No Longer Truncated in String Functions

SQL Server releases 11.0 and earlier truncated the trailing spaces of the input *char* value before performing string operations. Adaptive Server does not.

The following simple example uses the **upper** string function to demonstrate the effects of this change:

```
declare @c1 char(10)
select @c1 = 'abc'
select upper(@c1)
```

Pre-11.5 releases produce a *char*(3) with a value of “ABC”. Adaptive Server produces a *char*(10) with a value of “ABC#######”, where each # represents a trailing blank.

The following example also demonstrates this change in behavior:

```
select 'x'+upper(convert(char(5),'abc'))+'x'
```

Pre-11.5 releases produced a *char*(5): "xABCx". Adaptive Server produces a *char*(7): "xABC x".

Convert Function No Longer Supports Precision

Pre-11.5 releases accepted two parameters when converting a *numeric* or *decimal* datatype to a character datatype: *length* and *precision*. Adaptive Server does not.

For example, suppose you run the following conversion function on the *titles* table in the *pubs3* database, where the *price* column was created using the *money* datatype. A similar query in previous releases:

```
select convert(char(19,4), price) from titles
```

converts a *money* value to a *char* value.

In release 11.5, that query generates the following error message:

```
Msg 2716: Level 16, State2: Can't specify a length
or scale on type 'char'.
```

Union of *varbinary* Data No Longer Eliminates Incorrect Duplicates

Pre-11.5 releases incorrectly evaluated different *varbinary* data as duplicate. Adaptive Server does not.

For example, assume that you create a union between a *binary(30)* column and a *varbinary(255)* column. In previous releases, the resulting column type is *binary(30)*. Therefore, any row that is identical for the first 30 bytes is dropped as a duplicate.

In release 11.5, the resulting column is of type *varbinary(255)*. Therefore, the result values are not dropped as duplicates.

Result Differences in Parallel Queries

When a query does not include vector or scalar aggregates or does not require a final sorting step, a parallel query might return results in a different order from the same query run in serial, and subsequent executions of the same query in parallel might return results in different order each time.

Results from serial and parallel queries that include vector or scalar aggregates, or require a final sort step, are returned after all of the

results from worktables are merged or sorted in the final query processing step. Without query clauses that require this final step, parallel queries send results to the client using a network buffer merge, that is, each worker process sends results to the network buffer as it retrieves the data that satisfies the queries.

The relative speed of the different worker processes leads to the differences in result set ordering. Each parallel scan behaves a little differently, due to pages already in cache, lock contention, and so forth. Parallel queries always return the same **set** of results, just not in the same **order**. If you need a dependable ordering of results, you must use **order by** or run the query in serial mode.

In addition, due to the pacing effects of multiple worker processes reading data pages, two types of queries may return different results accessing the same data when an aggregate or a final sort is not done:

- Queries that use **set rowcount**
- Queries that select a column into a local variable without sufficiently restrictive query clauses

Queries That Use *set rowcount*

The **set rowcount** option stops processing after a certain number of rows are returned to the client. With serial processing, the results are consistent in repeated executions. In serial mode, the same rows are returned in the same order for a given **rowcount** value, because as single process reads the data pages in the same order every time.

With parallel queries, the order of the results and the set of rows returned can differ, due to the fact that worker processes may access pages sooner or later than other processes. When **set rowcount** is in effect, each row is written to the network buffer as it is found and the buffer is sent to the client when it is full, until the required number of rows have been returned. To get consistent results, you must either use a clause that performs a final sort step or run the query in serial mode.

Queries That Set Local Variables

This query sets the value of a local variable in a **select** statement:

```
select @tid = title_id from titles
    where type = "business"
```

The **where** clause in the query matches multiple rows in the *titles* table. so he local variable is always set to the value from the last matching

row returned by the query. The value is always the same in serial processing, but for parallel query processing, the results depend on which worker process finishes last. To achieve a consistent result, use a clause that performs a final sort step, execute the query in serial mode, or add clauses so that the query arguments select only single rows.

Achieving Consistent Results

To achieve consistent results for the types of queries discussed in this section, you can either add a clause to enforce a final sort or you can run the queries in serial mode. The query clauses that provide a final sort are:

- **order by**
- **distinct**, except for uses of **distinct** within an aggregate, such as **avg(distinct price)**
- **union**, but not **union all**

To run queries in serial mode, you can:

- Use **set parallel_degree 1** to limit the session to serial operation

Include the **(parallel 1)** clause after each table listed in the **from** clause of the query.

3 New Features and System Changes in SQL Server Release 11.0

This chapter describes the features and system changes introduced in SQL Server release 11.0. If you are upgrading to release 11.5 from a pre-11.0 release, these features will be new to you.

Topics covered are:

New Features in Release 11.0

The new features in release 11.0 are as follows:

User-Defined Caches

Release 11.0 allows System Administrators to split the SQL Server data cache into separate named data caches and to bind databases or

database objects to those caches. Also, System Administrators can create pools within caches that perform large I/O to disk, improving performance for many queries.

Configuring Caches

Named data caches are created with the new system procedure **sp_cacheconfig**. Memory pools for large I/O are configured within data caches with **sp_poolconfig**. Configuring caches and pools can also be done by editing a configuration file.

Entities are bound to caches with **sp_bindcache** and unbound with **sp_unbindcache** or **sp_unbindcache_all**. The system procedure **sp_helpcache** provides information about caches and cache bindings. **sp_cacheconfig** also provides information about caches and the pools within caches.

See Chapter 9, "Configuring Data Caches," in the *System Administration Guide* for instructions on configuring named caches and binding database objects to them.

Binding Objects to User-Defined Caches

In general, you can increase performance by binding the following objects to their own named caches:

- *sysindexes* table and its index

  Bind these two objects to the same named cache. This is the most frequently used table and its index. However, you do not have to make its named cache very large.

- *syslogs* table

  This helps reduce the contention on the buffer manager spinlocks. You should also configure a 4K memory pool in its named cache to benefit from the larger log I/O size, as defined by **sp_logiosize**.

- *tempdb* database

  This helps performance if many temporary or work tables are generated by your applications.

In addition to the above objects, the most frequently used tables and indexes (based on your database and application designs) are good candidates for binding to their own caches. However, if you have only one table with no index, binding that table to a named cache will not improve performance.

➤ ***Note***

Having too many named caches without increasing the total memory available to your server is generally not a good idea. Each additional named cache reduces the amount of memory available in the default cache.

Cache Strategies

SQL Server release 11.0 provides new optimization strategies and new commands to control the use of those strategies for objects, sessions, and queries. Figure 3-1 shows the two cache strategies.

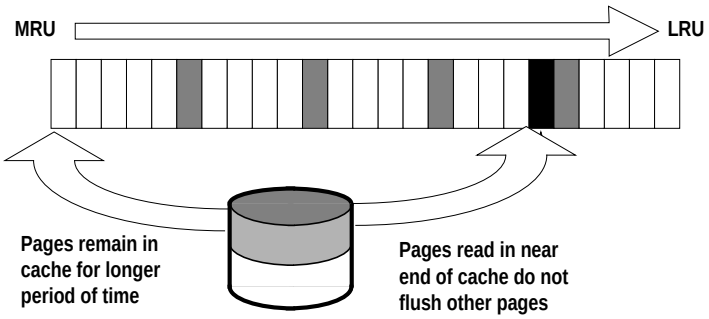

Figure 3-1: Cache strategy choices

SQL Server can:

- Read pages in at the start of the MRU (most recently used) chain in order to keep the pages in cache so that they can be accessed many times without performing additional I/O
- Read pages in near the end of the cache so that they will not force other pages out of the cache

See Chapter 16, "Memory Use and Performance," in the *Performance and Tuning Guide* for performance information on cache strategies.

Large I/O

When caches are created, all space is assigned to a 2K pool. This pool can be split to allow the cache to perform large I/O, reading up to eight data pages at once. Since most I/O time is spent performing queueing, seeking, and positioning operations, large I/O can greatly

increase performance for queries that scan entire tables or ranges of tables.

SQL Server can also perform large I/O on the transaction log.

Changing the Log I/O Size

Using the new **sp_logiosize** system procedure, you can change the log I/O size used by SQL Server. This can improve performance by reducing the number of times that SQL Server writes transaction log pages to disk. The value you specify for **sp_logiosize** must correspond to an existing memory pool configured for the cache used by the database's transaction log.

By default, SQL Server defines the log I/O size of a database as 4K. If the transaction log for a database is bound to the default cache or a user-defined cache that does not contain a 4K memory pool, SQL Server sets the log I/O size to 2K (a 2K memory pool is always present in any cache). For most workloads, a log I/O size of 4K performs much better than one of 2K, so each cache used by a transaction log should have a 4K memory pool.

New Query Tuning Options

New query tuning options support the capabilities of named caches and provide additional control when tuning queries:

- The **set forceplan on** command forces SQL Server to join the tables in a query in the order in which they are listed in the **from** clause.
- The **set table count** command allows you to specify the number of tables that are considered at once as joins are optimized.
- The new system procedure **sp_cachestrategy** disables and reenables cache strategies and large I/O for individual objects.
- New clauses for **select**, **delete**, and **insert** commands allow you to specify the index, cache strategy, or I/O size for a query.
- The **set prefetch** command allows you to enable or disable large I/O for a session.

See Chapter 10, "Advanced Optimizing Techniques," in the *Performance and Tuning Guide* for information on query processing options that control caching strategies and I/O size.

Data Storage Changes

This section describes changes to the way SQL Server stores data.

Maximum Number of Rows per Page

You can now configure the maximum number of rows stored on a data page or index leaf page. The **max_rows_per_page** value specified in a **create table**, **create index**, or **alter table** command restricts the number of rows allowed on a data page, a clustered index leaf page, or a nonclustered index leaf page. This reduces lock contention and improves concurrency for frequently accessed tables.

The **max_rows_per_page** value applies to the data pages of an unindexed table or the leaf pages of an index. The value of **max_rows_per_page** is stored in the *maxrowsperpage* column of *sysindexes*. (The *maxrowsperpage* column was named *rowpage* in previous releases of SQL Server.)

Unlike **fillfactor**, which is not maintained after creating a table or index, SQL Server retains the **max_rows_per_page** value when adding or deleting rows. For information about how to use **max_rows_per_page** to reduce lock contention and improve concurrency for your server, see the *Performance and Tuning Guide*.

Page Allocation

SQL Server 10.0 and earlier releases searched the OAM page chain for unused pages before allocating new extents to objects. Release 11.0 provides a system-wide configuration parameter, **page utilization percent**, that allows SQL Server to allocate a new extent to an object without searching the OAM page chain, depending on the ratio of used and unused pages in the table to the table's allocated extents.

Partitioned Tables

➤ ***Note***

11.5 Note: Release 11.5 adds the ability to scan partitioned tables in parallel and to create clustered indexes.

By default, SQL Server stores a **heap table**'s data in one double-linked chain of database pages. When a transaction inserts a row into the table, it holds an exclusive page lock on the last page of the page

chain while inserting the row. As multiple transactions attempt to insert rows into the same table at the same time, performance problems can occur. Because only one transaction at a time can obtain an exclusive lock on the last page, other, concurrent insert transactions block, as shown in Figure 3-2.

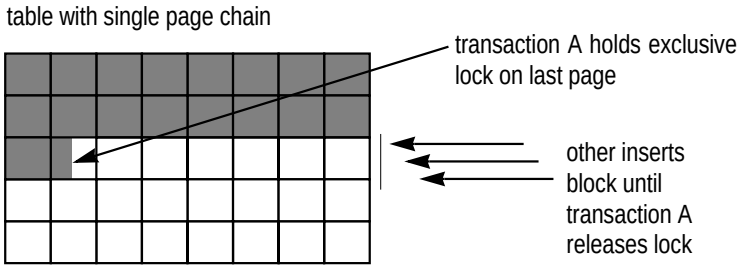

Figure 3-2: Page contention during inserts

In release 11.0, SQL Server provides the ability to partition heap tables. A **partition** is simply another term for a page chain. Partitioning a table creates multiple page chains (partitions) for the table and, therefore, multiple last pages for insert operations.

When a transaction inserts data into a partitioned table, SQL Server randomly assigns the transaction to one of the table's partitions. Concurrent inserts are less likely to block, since multiple last pages are available for inserts, as shown in Figure 3-3.

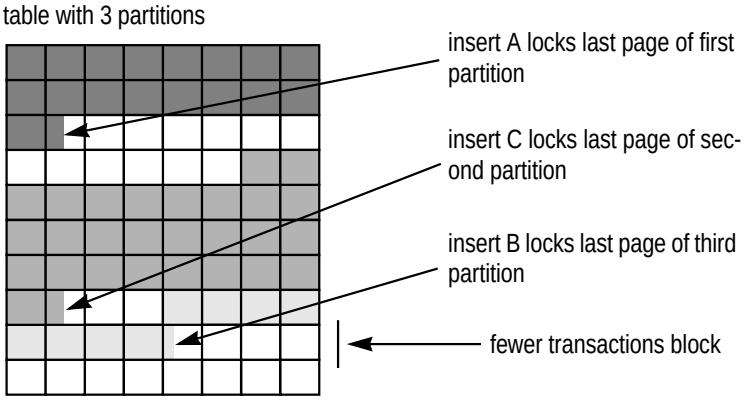

Figure 3-3: Addressing page contention with partitions

If a table's segment spans several physical disks, SQL Server distributes the table's partitions across those disks when you create the partitions. This can improve I/O performance when SQL Server writes the table's cached data to disk, since the I/O is distributed over several devices.

SQL Server release 11.0 manages partitioned tables transparently to users and applications. Partitioned tables appear exactly like unpartitioned tables, except when accessed via the **dbcc checktable** and **dbcc checkdb** commands or when viewed with the new **sp_helpartition** procedure.

See Chapter 17, "Controlling Physical Data Placement," of the *Performance and Tuning Guide* for information about how to create and administer partitioned tables.

Transaction Log Changes

This section describes the changes made to the transaction log for release 11.0.

User Log Caches

There is one user log cache for each configured user connection. SQL Server uses these user log caches to buffer the user transaction log records, which reduces the contention at the end of the transaction log. When a user log cache becomes full or when another event

occurs (such as when the transaction completes), SQL Server "flushes" all log records from the user log cache to the database transaction log. You can configure the size of all user log caches for your server using the **user log cache size** parameter of **sp_configure**; the default size is 2048 bytes.

Since more than one process can access the contents of a user log cache when determining a possible "flush," SQL Server uses spinlocks to protect the user log caches. A spinlock is an internal locking mechanism that prevents a second process from accessing the data structure being used by another process. SQL Server uses the **user log cache spinlock ratio** parameter of **sp_configure** to specify the ratio of user log caches per user log cache spinlock; the default is 20 user log caches for each spinlock.

For information about configuring **user log cache size** and **user log cache spinlock ratio**, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*.

syslogshold Table

You can query the new *syslogshold* system table to determine the oldest active transaction in each database. *syslogshold* exists in the *master* database, and each row in the table represents either:

- The oldest active transaction in a database or
- The Replication Server® truncation point for the database's log.

A database may have no rows in *syslogshold*, a row representing one of the above, or two rows representing both of the above. For information about how Replication Sever truncation points affects the truncation of a database's transaction log, see the Replication Server documentation.

Querying *syslogshold* can help you when the transaction log becomes too full, even with frequent log dumps. The **dump transaction** command truncates the log by removing all pages from the beginning of the log up to the page that precedes the page containing an uncommitted transaction record (the oldest active transaction). The longer this active transaction remains uncommitted, the less space is available in the transaction log, since **dump transaction** cannot truncate additional pages.

For information about how to query *syslogshold* to determine the oldest active transaction that is holding up your transaction dumps, see Chapter 21, "Backing Up and Restoring User Databases," in the *System Administration Guide*.

Isolation Level 0

You can specify isolation level 0 for the queries in transactions along with the isolation levels 1 and 3 supported in release 10.0. Isolation level 0 prevents other transactions from changing data that has already been modified by an uncommitted transaction. The other transactions are blocked from modifying that data until the transaction commits. However, other transactions can still read the uncommitted data (known as **dirty reads**).

Queries executing at isolation level 0 do not acquire any read locks for their scans, so they do not block other transactions from writing to the same data (and vice versa). Applications that are not impacted by dirty reads may see better concurrency and reduced deadlocks when accessing the same data using isolation level 0. However, transactions that require data consistency should **not** use isolation level 0.

You can specify isolation level 0 for the transactions of a session as follows:

```
set transaction isolation level 0
```

Since this command makes all queries execute at isolation level 0, you should be wary of using it for any transaction that requires data consistency. Instead, you can selectively choose isolation level 0 for a query in a transaction using the **at isolation** clause as follows:

```
select *
from titles
at isolation read uncommitted
```

For information about transactions and isolation levels, see Chapter 18, “Transactions: Maintaining Data Consistency and Recovery,” in the *Transact-SQL User’s Guide*.

Lock Manager Changes

This section describes the changes to locking behavior.

Table, Page, and Address Lock Tables

SQL Server manages the acquiring and releasing of table locks by using an internal hash table with 101 rows (known as **hash buckets**) and of page and address locks by using internal hash tables with 1031 rows each. In release 11.0, these tables use one or more spinlocks to serialize access between processes that are running on different

engines. A spinlock is an internal locking mechanism that prevents a second process from accessing the resource currently being used by another process. All processes trying to access the resource must wait (or **spin**) until the lock is released.

For SQL Servers running with multiple engines, you can set the ratio of spinlocks protecting each hash table using the **table lock spinlock ratio**, **page lock spinlock ratio**, and **address lock spinlock ratio** configuration parameters. The table locks hash table default ratio is 20 rows for each spinlock, and the page or address locks hash table default ratio is 100 rows for each spinlock. A SQL Server configured with only one engine has only one spinlock for each hash table, regardless of the values specified for these parameters.

For information about configuring the **table lock spinlock ratio**, **page lock spinlock ratio**, and **address lock spinlock ratio** parameters, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*.

Deadlock Checking

SQL Server performs deadlock checking after a minimum period of time for any process that is waiting (or sleeping) for a lock to be released. By default, this minimum period of time is 500 ms. Previous releases of SQL Server perform this deadlock check at the time the process begins to wait for a lock. This deadlock checking is a time-consuming overhead for applications that wait without a deadlock.

You can change the minimum amount of time (in milliseconds) that a process must wait before it initiates a deadlock check using the **deadlock checking period** configuration parameter. If you expect your applications to deadlock infrequently, you can delay deadlock checking even further and reduce the overhead cost. However, configuring **deadlock checking period** to a higher value produces longer delays before deadlocks are detected.

For information about deadlock checking, see Chapter 5, "Locking in Adaptive Server," in the *Performance and Tuning Guide*.

New Engine Freelock Lists

In release 11.0, when a process that is running on a multi-engine SQL Server requests a lock, it looks for one in its engine's freelock list. If the engine freelock list is out of locks, SQL Server moves a certain number of locks from its global freelock list to the engine freelock list.

For single-engine SQL Servers, the entire global freelock list is moved to the engine freelock list at server start-up time.

After an engine completes a process, all locks held by that process are released and returned to that engine's freelock list. This reduces the contention of each engine accessing the global freelock list. However, if the number of locks released to the engine exceed the maximum number of locks allowed in the engine's freelock list, SQL Server moves a number of locks to the global freelock list. This replenishes the number of locks that are available to other engines from the global list.

You can configure the maximum number of locks available to the engine freelock lists as a percentage of the total number of locks available to your server with the **max engine freelocks** configuration parameter. You can also configure the number of locks transferred back and forth between the engine and global freelock lists using the **freelock transfer block size** parameter. For information about these parameters, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*.

Increasing the Engine Freelock Lists

Each engine's freelock list is a portion of the global freelock list allocated to the engines, defined by the **max engines freelocks** configuration parameter as a percentage of the total number of locks. By default, SQL Server assigns 10 percent of the total number of locks to all the engines' freelock lists; the remainder stays in the global freelock list. For example, if your server is configured with 5000 locks, 500 locks are distributed evenly between each engine freelock list, and 4500 remain in the global freelock list.

For most SQL Server configurations, you can double the percentage of locks assigned to the engine freelock lists and improve the performance of your applications.

Housekeeper Task

When SQL Server has no user tasks to process, a housekeeper task automatically begins writing dirty buffers from cache to disk. Because these writes are done during the server's idle cycles, they are known as **free writes**.

The benefits of the housekeeper task are:

- Improved CPU utilization

- Decreased need for buffer washing during transaction processing
- Faster checkpoints
- Shorter recovery time

In applications that repeatedly update the same database page, the housekeeper task may initiate some database writes unnecessarily. System Administrators can use the **housekeeper free write percentage** configuration parameter to disable the housekeeper task or to control its side effects.

For more information about the housekeeper task, see Chapter 21, "How Adaptive Server Uses Engines and CPUs," in the *Performance and Tuning Guide*.

SQL Server Configuration

In previous releases, SQL Server was configured using **sp_configure**, **reconfigure**, and a number of undocumented **dbcc tune** and **buildmaster -y** parameters. In order to provide a single entry point for all SQL Server configuration (except for Buffer Manager), **sp_configure** has been redesigned to incorporate the configuration parameters previously implemented with **dbcc tune** and **buildmaster -y**. In addition, a number of new configuration parameters have been implemented.

For information about which old configuration names have new names, see "New Names for Existing Configuration Parameters" on page 3-25.

The **sp_configure** interface has been redesigned to include these new features:

- Display levels, which allow the user to select one of three views of the configuration parameters:
  - Basic level, appropriate for general tuning of SQL Server
  - Intermediate level, somewhat more inclusive than the Basic level
  - Comprehensive level, appropriate for the most detailed level of SQL Server tuning
- Parameter hierarchy, which organizes the configuration parameters according to the area of SQL Server behavior to which they pertain.
- Configuration files, which allow users to replicate specific configurations, validate configurations before setting them, and create multiple configurations that can be easily switched.

For information about SQL Server configuration, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide.*

Lock Escalation

In previous releases of SQL Server, once a statement accumulated more than 200 page locks on a table, SQL Server tried to issue a table lock on that object. If the table lock succeeded, the page locks were no longer necessary and were released.

The **lock promotion HWM**, **lock promotion LWM**, and **lock promotion PCT** configuration parameters and the **sp_setpglockpromote** system procedure configure the number of page locks that SQL Server acquires on a table before it attempts to escalate to a table lock on a server-wide, per database, and per table basis.

For more information, see Chapter 5, "Locking in Adaptive Server," in the *Performance and Tuning Guide*.

Multiple Network Engines

If your symmetric multiprocessing (SMP) system supports network affinity migration, each SQL Server engine handles the network I/O for its connections in release 11.0. During login, SQL Server migrates the client connection task from engine 0 to the engine that is servicing the smallest number of connections. The client's tasks run network I/O on that engine until the connection is terminated.

By distributing the network I/O among its engines, SQL Server can handle more user connections. The per-process limit on the maximum number of open file descriptors no longer limits the number of connections. Adding more engines linearly increases the maximum number of file descriptors. For more information about how multiple network engines work in an SMP environment, see Chapter 10, "Managing Multiprocessor Servers," in the *System Administration Guide.*

Improvements to *showplan*

The output for the **set showplan** command has been improved as follows:

- Indents and delimiters have been added to help readability.

- Line numbers and statement numbers help debug long batches and procedures.
- Additional messages include the keys used on indexes and messages that provide more information about access methods.
- Subquery types, nesting levels, and other subquery information help check subquery performance.
- Messages have been added for the new release 11.0 features to display the I/O size and the caching strategy.

In the following example, there is an index named *title_ix* on the *titles* table:

```
select title_id, price
    from titles
    where title = "Computers and Privacy"
```

This is the pre-11.0 output from **showplan**:

```
STEP 1
The type of query is SELECT.
FROM TABLE
titles
Nested iteration
Index : title_ix
```

This is the release 11.0 output from **showplan**:

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


    STEP 1
        The type of query is SELECT.

        FROM TABLE
            titles
        Nested iteration.
        Index : title_ix
        Ascending scan.
        Positioning by key.
        Keys are:
            title
        Using I/O Size 2 Kbytes.
        With LRU Buffer Replacement Strategy.
```

All **showplan** messages are documented in Chapter 9, "Understanding Query Plans," of the *Performance and Tuning Guide*.

Query and Data Modification Changes

This section describes the changes made to subquery processing and update behavior.

Subquery Changes

The behavior of certain subqueries has changed. See “Changes That May Affect Existing Applications” on page 3-24 for detailed information about these changes and how they may affect your existing applications.

Update Changes

Update strategies for SQL Server have improved. More updates are now made in place or directly, with fewer writes to the transaction log. See the *Performance and Tuning Guide* for more information on performance and update operations.

Upgrading Database Dumps

When a SQL Server installation is upgraded to a new release, all databases associated with that server are automatically upgraded.

As a result, the database and transaction log dumps created with any previous SQL Server must be upgraded before they can be used with the current SQL Server.

SQL Server release 11.0 provides an automatic upgrade mechanism, on a per-database basis, for upgrading a database or transaction log dump from any SQL Server release 10.0 to the current SQL Server, thus making the dump compatible for use. This mechanism is entirely internal to SQL Server release 11.0 and requires no external programs. It provides the flexibility of upgrading individual dumps as needed.

The following tasks are not supported by this automatic upgrade functionality:

- Loading a SQL Server release 10.0 *master* database onto release 11.0.
- Installing new or modified stored procedures. Continue to use **installmaster.**
- Loading and upgrading dumps prior to SQL Server release 10.0.

Associated with this feature is a new **online database** command. This command marks a database available for public use after a normal load sequence, and if needed, upgrades a loaded database and/or transaction log dumps to the current version of SQL Server.

For more information on upgrading database dumps, see Chapter 21, "Backing Up and Restoring User Databases," in the *System Administration Guide* and the **load database** and **dump database** commands in the *Reference Manual.*

For more information on the **online database** command, see the *Reference Manual.*

Tape Device Determination by Backup Server

To become less device dependent and more flexible, Backup Server™ provides a method for determining the tape device characteristics (tape positioning for read, close, append, I/O size, file marks, and ability to overwrite a tape mark) for a dump operation.

When you issue a **dump database** or **dump transaction** command, Backup Server checks to see whether the device type of the specified dump device is known (supplied and supported internally) by SQL Server. If the device is not a known type, Backup Server checks the tape configuration file (default location is *$SYBASE/backup_tape.cfg)* for the device configuration. If the configuration is found, the **dump** command proceeds.

If the configuration is not found in the tape device configuration file, the **dump** command fails with the following error message:

```
Device not found in configuration file. INIT needs
to be specified to configure the device.
```

This means the device needs to be configured. Issue the **dump database** or **dump transaction** with **init** qualifier to configure the device. Using operating system calls, Backup Server attempts to determine the device's characteristics, and if successful, stores the device characteristics in the tape configuration file.

If Backup Server cannot determine the dump device characteristics, it defaults to one dump per tape. The device can not be used if the configuration fails to write at least one dump file.

Tape configuration by Backup Server applies only to UNIX platforms.

For more information on tape device determination and the tape configuration file, see Chapter 21, "Backing Up and Restoring User Databases," in the *System Administration Guide.*

IDENTITY Column Changes

The IDENTITY column feature allows you to create a column with system-generated values that uniquely identify each row in a table. Release 11.0 provides the following enhancements to IDENTITY columns:

- For tables with no unique indexes, the **identity in nonunique index** database option allows you to create a unique index by including an IDENTITY column in the index key. Unique indexes are required for isolation level 0 reads and updatable cursors.
- The **identity grab size** configuration parameter allows each SQL Server process in a multiprocessor environment to reserve a specified number of IDENTITY column values when you add rows to tables that contain IDENTITY columns.
- The **size of auto identity** configuration parameter sets the precision of the IDENTITY columns that SQL Server generates for tables created in databases where the **auto identity** database option is set to **true**.

For more information about the enhancements to IDENTITY columns, see Chapter 7, "Creating Databases and Tables," in the *Transact-SQL User's Guide*.

New *text* and *image* Global Variables

The following five global variables have been added to release 11.0 to facilitate inserting or updating *text* and *image* data through Open Client™ applications:

- *@@textcolid*
- *@@textdbid*
- *@@textobjid*
- *@@textptr*
- *@@textts*

For information about these global variables, see Chapter 13, "Using Batches and Control-of-Flow Language," in the *Transact-SQL User's Guide*. For information about how to use these variables in a

procedure for Open Client applications, see the *Client-Library/C Reference Manual*.

System Changes in Release 11.0

This section provides a summary of release 11.0 changes. The changes are:

New *online database* Command

Table 3-1 describes the new **online database** command:

Table 3-1: New online database command

Name	Function
online database	Makes a database available for public use after a normal load sequence and, if needed, upgrades a loaded database and transaction log dumps to the current version of SQL Server.

Changes to Existing Commands

Table 3-2 summarizes the changes to existing commands:

Table 3-2: Changes to existing commands

Name	Change
alter table	The new **partition** clause allows you to create additional page chains for a table with no clustered index. The new **unpartition** clause allows you to concatenate all page chains for a partitioned table.
buildmaster	The **buildmaster -r** flag is no longer supported.
delete	New clauses allow specifying the index, cache strategy, and I/O size for a query.

Table 3-2: Changes to existing commands (continued)

Name	Change
reconfigure	The **reconfigure** command is no longer required to activate changes made with **sp_configure**. **reconfigure** is now non-operational. Scripts using **reconfigure** will still run, but should be changed at your earliest convenience, as **reconfigure** may not be supported in future releases.
select	New clauses allow specifying the index, cache strategy, and I/O size for a query. The new **at isolation** clause allows you to specify the isolation level for the query.
set showplan	Changes to output.
set transaction isolation level	New isolation level option: 0
update	New clauses allow you to specify the index, cache strategy, and I/O size for a query.

New *set* options

Table 3-3 summarizes the new **set** options:

Table 3-3: New set options

Option	Function
forceplan	Forces the optimizer to perform joins in the order in which tables are named in the **from** clause of a query.
prefetch	Disables or enables large I/O (data prefetch) for a session.
statistics subquerycache	Prints statistics about the use of the subquery cache during the execution of a subquery.
table count *integer*	Specifies the number of tables that are to be optimized at the same time in a join query.

New System Procedures

The following new system procedures provide some of the functionality for the features described earlier:

Table 3-4: New system procedures

Procedure	Function
sp_bindcache	Binds databases, tables, indexes, or *text* or *image* chains to data caches.
sp_cacheconfig	Configures named data caches and provides information about caches.
sp_cachestrategy	Enables and disables caching strategies and large I/O for specific tables and indexes.
sp_chgattribute	Changes the **max_rows_per_page** value for future space allocations of a table or index.
sp_displaylevel	Sets and displays a user's display level. The display level determines which SQL Server configuration parameters are displayed in **sp_configure** output.
sp_dropglockpromote	Removes lock promotion values from a table or database.
sp_helpcache	Provides information about cache overhead requirements and provides information about caches and cache bindings.
sp_helpartition	Lists the first page and the control page for each partition of a partitioned table.
sp_logiosize	Changes the log I/O size used by SQL Server to a different memory pool when doing I/O for the transaction log of the current database.
sp_poolconfig	Configures pools within named caches to enable large I/O.
sp_procqmode	Reports the subquery processing mode of an object.
sp_setpglockpromote	Sets or changes the lock promotion thresholds for a database, table, or for SQL Server.
sp_unbindcache	Unbinds a specific database, table, index, or *text* or *image* object from a data cache.
sp_unbindcache_all	Unbinds all objects bound to a cache.

Changes to System Procedures

The following system procedures have been changed in this release:

Table 3-5: Changes to system procedures

Procedure Name	Change
sp_dboption	Has a new database option, **identity in nonunique index,** that allows you to create a unique index by including an IDENTITY column in the index key.
sp_configure	Includes new parameters and new optional subcommands, and displays new output.

New System Tables

The following system tables are new in this release:

Table 3-6: New system tables

Table Name	Function
sysattributes	Defines attributes for objects such as databases, tables, indexes, users, logins, and procedures.
syslogshold	Stores information about the oldest active transaction and the Replication Server truncation point for each database
syspartitions	Stores internal information about table partitions.

Changes to Existing System Tables

Table 3-7 lists system tables that have been changed in this release:

Table 3-7: Changes to system tables

Table Name	Change
sysindexes	Changed column: *rowpage* to *maxrowsperpage* The *root* column entry for a table becomes obsolete when the table is partitioned. Root pages for individual partition are derived from information in the new *syspartitions* table.

Table 3-7: Changes to system tables (continued)

Table Name	Change
sysconfigures	Added new columns: • *name* contains the configuration parameter name • *parent* contains the group to which the parameter belongs • *value2* is used for configuration parameters whose values are character strings • *value3* is not currently used
syscurconfigs	Added new columns: • *value2* is used for configuration parameters whose values are character strings • *defvalue* contains default values for configuration parameters • *minimum_value* contains the minimum legal value for a configuration parameter • *maximum_value* contains the maximum legal value for a configuration parameter • *memory_used* contains the per-unit amount of memory used by a configuration parameter • *display level* contains the display level associated with a configuration parameter • *datatype* and *message_num* are for internal use
sysdatabases	Added new column: • *offline* status bit is used by the upgrading database dumps feature

Changes That May Affect Existing Applications

This section describes the system changes introduced in release 11.0 that may affect your applications if you are upgrading to release 11.5 from a pre-11.0 release. Topics covered are as follows:

New Transact-SQL Keywords in Release 11.0

The following keywords are new reserved words in SQL Server 11.0. They cannot be used as object names or column names.

- **errordata**
- **max_rows_per_page**
- **online**
- **partition**
- **unpartition**

You must change all database names that are new reserved words before you can upgrade an earlier version of the server. You can change table, view, and column names or use delimited identifiers. Once you upgrade, you cannot use database objects whose names are new reserved words until you modify your procedures, SQL scripts, and applications.

Changes to SQL Server Configuration

The following changes to the SQL Server configuration interface will affect existing applications.

The *reconfigure* Command

In previous releases, changes made with **sp_configure** had to be activated with the **reconfigure** command. This is no longer required. **reconfigure** is no longer operational. Existing scripts should continue to run, but should be changed at your earliest convenience because **reconfigure** may not be supported in future releases.

buildmaster -r No Longer Supported

The **buildmaster -r** flag, which was used to rewrite the configuration block with default values for SQL Server configuration parameters, is no longer supported.

To run SQL Server with its built-in default values:

1. Rename, move, or delete your configuration file.
2. Shut down and restart SQL Server.

When you do this, SQL Server uses the built-in default configuration values, and creates a configuration file *server_name.cfg* in the directory from which SQL Server was started. However, these values do not get written to *sysconfigures*. In order to have them written to *sysconfigures*, you need to shut down and restart SQL Server once again.

New Names for Existing Configuration Parameters

The following SQL Server configuration parameters (formerly called "configuration variables") have new names:

Table 3-8: New configuration parameter names for release 11.0

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

Table 3-8: New configuration parameter names for release 11.0 (continued)

Old Name	New Name
allow updates	allow updates to system tables
calignment	memory alignment boundary
cclkrate	sql server clock tick length
cfgcprot	permission cache entries
cguardsz	stack guard size
cindextrips	number of index trips
cmaxnetworks	max number network listeners
cmaxscheds	i/o polling process count
cnalarm	number of alarms
cnblkio	disk i/o structures
cnlanginfo	number of languages in cache
cnmaxaio_engine	max async i/os per engine
cnmaxaio_server	max async i/os per server
cnmbox	number of mailboxes
cnmsg	number of messages
coamtrips	number of oam trips
cpreallocext	number of pre-allocated extents
cpu flush	cpu accounting flush interval
cschedspins	runnable process search count
csortbufsize	number of sort buffers
csortpgcount	sort page count (Removed in release 11.5.)
ctimemax	cpu grace time
database size	default database size
default language	default language id
devices	number of devices
extent i/o buffers	number of extent i/o buffers (Removed in 11.5.)
fillfactor	default fill factor percent
i/o flush	i/o accounting flush interval
language in cache	number of languages in cache
locks	number of locks

Table 3-8: New configuration parameter names for release 11.0 (continued)

Old Name	New Name
maximum network packet size	max network packet size
memory	total memory
mrstart	shared memory starting address
nested trigger	allow nested triggers
open databases	number of open databases
open objects	number of open objects
password expiration interval	systemwide password expiration
pre-read packets	remote server pre-read packets
procedure cache	procedure cache percent
recovery flags	print recovery information
recovery interval	recovery interval in minutes
remote access	allow remote access
remote connections	number of remote connections
remote logins	number of remote logins
remote sites	number of remote sites
sql server code size	executable code size
tape retention	tape retention in days
user connections	number of user connections

New Configuration Parameters

Release 11.0 introduced a number of new configuration parameters listed in Table 3-9. See Chapter 11, “Setting Configuration

Parameters," in the *System Administration Guide* for a complete description of these parameters.

Table 3-9: New SQL Server configuration parameters for release 11.0

address lock spinlock ratio
configuration file
deadlock checking period
deadlock retries
event buffers per engine
freelock transfer block size
housekeeper free write percent
identity grab size
lock promotion HWM
lock promotion LWM
lock promotion PCT
max engine freelocks
o/s async i/o enabled
o/s file descriptors
page lock spinlock ratio
page utilization percent
partition groups
partition spinlock ratio
size of auto identity column
table lock spinlock ratio
total data cache size
user lock cache size
user lock cache spinlock ratio

New *deadlock checking period* Parameter

Previous releases of SQL Server performed deadlock checking when the process began to wait for a lock. In release 11.0, SQL Server performs deadlock checking for a process only after it has waited a

minimum of 500 ms. for a lock to be released. To return to the previous way of deadlock checking, set **deadlock checking period** to 0.

New *page utilization percent* Parameter

The default behavior of previous releases of SQL Server is to search the OAM page chain for unused pages before allocating a new extent. To keep this behavior, set the **page utilization percent** to 100.

Compiled Object Sizes Have Grown

The size of compiled objects is larger in System 11, and was significantly larger in System 10. You may need to resize your procedure cache to maintain the same performance.

SQL Server Code Size Has Grown

More memory is used for the kernel and other internal structures. You may need to add more memory to maintain the same performance as your previous release.

Subquery Changes

Several problems relating to subqueries have been fixed in SQL Server release 11.0. This section describes old and new behavior in detail. Examine applications that use subqueries to determine if your application relies on behavior that has been corrected.

Subquery processing in previous releases of SQL Server used "inside-out" processing. Release 11.0 subqueries are processed "outside-in" for greater efficiency. Upgrading to release 11.0 does not automatically change the processing mode of the subquery.

After upgrading to release 11.0, you must drop and re-create objects to take advantage of the new processing style. Use **sp_procqmode** to identify objects that include subqueries. After upgrading to SQL Server release 11.0, you cannot create an object that uses release 10.0 processing.

It is best to test procedures that contain subqueries and make any changes that are necessary for your application before upgrading your production system.

Different Results

Subqueries in objects created in SQL Server release 11.0 may return different results than subqueries in objects created on an earlier SQL Server release. To test performance after upgrade, create the objects in SQL Server release 11.0, test the results and performance, and make any changes that are necessary for your application.

Determining Query Processing Mode

A new system procedure, **sp_procqmode**, reports on the subquery processing mode of an object.

Duplicates in Subquery Results

As of release 10.0, subqueries using **in** or **any** predicates no longer return duplicates.

set dup_in_subquery on was provided as an upgrade path to SQL Server release 10.0. It is no longer supported. If your application depends on the duplicate rows in the result set, rewrite the subquery as a join.

For example:

```
select a from s where b in
    (select c from t)
```

would become:

```
select a from s, b where s.b = t.c
```

Procedures created in SQL Server release 10.0 that use **dup_in_subquery** will continue to run until the procedure is dropped and re-created in release 11.0.

Changes in Subquery Restrictions

The following subquery restrictions have been removed:

- You can use **distinct** with **group by** in a subquery.

  The **distinct** keyword suppressed duplicate rows in the output of pre-release 10.0 subqueries. This keyword is no longer necessary, and is ignored, in subqueries.

- You can reference a correlated variable in the select list of a subquery. For example, the following expression subquery is now supported:

  ```
  select t1.c1 from t1
      where t1.c2 = (select t2.c2 + t1.c3 from t2)
  ```

The following subquery restriction has been added:

- There is a maximum of 16 subqueries within a single side of a union.

Handling NULL Results

Prior to release 11.0, a correlated expression subquery in the **set** clause of an update returned 0 instead of NULL when there were no matching rows. Release 11.0 correctly returns NULL when there are no matching rows, and an error is raised if the column does not permit nulls.

If you have applications that depend on the pre-release 11.0 behavior, you will 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)
```

or:

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

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

Another example of this is the *totalsales_trig* in the *pubs2* sample database. In previous versions, the trigger read as follows:

```
create trigger totalsales_trig
on salesdetail
for insert, update, delete
as
/* Save processing:  return if there are no rows affected */
if @@rowcount = 0
        begin
                return
        end

/* add all the new values */
/* use isnull:  a null value in the titles table means
**              "no sales yet" not "sales unknown"
*/
update titles
        set total_sales = isnull(total_sales, 0) +
        (select sum(qty)
                from inserted
                where titles.title_id = inserted.title_id)

/* remove all values being deleted or updated */
update titles
        set total_sales = isnull(total_sales, 0) -
        (select sum(qty)
                from deleted
                where titles.title_id = deleted.title_id)
```

sum(qty) is NULL if no row is returned from the table, so when a statement changes the *total_sales* column, the trigger changes to NULL all the rows in *titles* that do not qualify.

To guarantee that the subquery in the expression for the update returns a non-null value, the corrected trigger is:

```
create trigger totalsales_trig
on salesdetail
for insert, update, delete
as
/* Save processing:  return if there are no rows affected */
if @@rowcount = 0
        begin
                return
        end

/* add all the new values */
/* use isnull:  a null value in the titles table means
```

```
**              "no sales yet" not "sales unknown"
*/
update titles
        set total_sales = isnull(total_sales, 0) +
        (select sum(qty)
                from inserted
                where titles.title_id = inserted.title_id)
        where title_id in (select title_id from inserted)

/* remove all values being deleted or updated */
update titles
        set total_sales = isnull(total_sales, 0) -
        (select sum(qty)
                from deleted
                where titles.title_id = deleted.title_id)
        where title_id in (select title_id from deleted)
```

Improved Performance

Most subqueries will perform better after being dropped and re-created.

However, an expression subquery containing an aggregate where the outer table is very large and has few duplicate correlation values, and the inner table is small, may not perform as well in release 11.0. For example:

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

The workaround is:

```
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.x
    and huge_table.y=#t.y
```

Changes to *showplan* Output in Release 11.0

The output for the **set** option **showplan** has changed. If you have applications that use this generated output, they need to be changed to use the new output.

See Chapter 9, "Understanding Query Plans" in the *Performance and Tuning Guide*.

New Caching Strategies May Affect Performance

The optimizer may now choose a new caching strategy called "fetch and discard" strategy. Standard caching strategy reads pages in at the head of the MRU / LRU (most recently used / least recently used) chain in cache. This flushes other pages out of cache. Fetch-and-discard (or MRU) strategy reads pages into cache much closer to the end of the cache, so it does not flush other pages. But these pages remain in cache a much shorter time, so subsequent user queries that might get a "cache hit" by finding these pages in cache under the standard strategy will be less likely to find the page in cache.

When the optimizer estimates that a particular query will read a significant number of pages, and only need those pages once during a query, it may choose fetch and discard strategy. If your disk I / O increases in release 11.0, check the I / O strategy for queries using **showplan** to see the cache strategy in use. If the queries are using fetch-and-discard (MRU) strategy, you can add a clause to **select, update,** and **delete** commands to specify standard caching.

See Chapter 10, "Advanced Optimizing Techniques" in the *Performance and Tuning Guide*.

Upgrading Database Dumps

If you use scripts to perform database loads using the **load database** and **load transaction** commands, the scripts will fail unless you include the **online database** command at the end of your load sequence. You should also remove the following **sp_dboption** options from your scripts: **no chkpt on recovery, dbo use only,** and **read only.** These options are no longer needed with release 11.0, as **load database** sets the database status to offline, thus making it inaccessible by users.

For more information on upgrading database dumps, see Chapter 21, "Backing Up and Restoring User Databases," in the *System*

Administration Guide and the **load database** and **dump database** commands in the *Reference Manual.*

Partitions and Physical Data Placement

➤ ***Note***

11.5 Note: Release 11.5 adds the ability to create clustered indexes on partitioned tables and to access the tables in parallel. See the *Performance and Tuning Guide* for more information.

In release 10.0, a heap table's data was always inserted at the end of a single page chain. This meant that a heap table (for example, a history table) would physically store its newest entries at the end of the page chain, and its oldest entries at the beginning. A simple **select** statement or cursor scan against such a table could return rows in roughly the same order in which they were inserted.

In release 11.0 you can create partitioned tables that have multiple page chains. Inserts into partitioned tables can occur at the ends of many separate page chains. Data in such a table **is not** physically stored in sequential order. If you want to view rows from a partitioned table in sequential order, use the **order by** clause in your **select** statement.

See the *Performance and Tuning Guide* for more information about partitions.

4 New Features and System Changes in SQL Server Release 10.0

This chapter describes the features and system changes introduced in SQL Server release 10.0. If you are upgrading to release 11.5 from a pre-10.0 release, these features will be new to you.

Topics covered are:

New Features in Release 10.0

The new features in release 10.0 are:

New Installation and Upgrade Utility

➤ ***Note***

The **sybinit** installation program for UNIX platforms is replaced by **sybsetup** in release 11.5. See "New Installation and Upgrade Utilities for UNIX" on page 2-1.

A new installation program, **sybinit**, provides a flexible, menu-driven tool for managing configuration of SQL Server and Backup Server. It can be used interactively or with a resource file, if you need to install or upgrade many servers.

sybinit allows you to:

- Install a new release 10.0 SQL Server or Backup Server
- Upgrade an existing SQL Server from release 4.8 or later to release 10.0
- Modify an existing SQL Server 10.0 to install and activate auditing
- Install languages and character sets
- Configure your default language, sort order, or character set
- Install other System 10™ products

It also helps manage your system's interfaces file, providing a convenient tool for adding, deleting or changing interfaces file entries.

See your SQL Server installation and configuration guide for information on running **sybinit.**

Addition to System Databases

Release 10.0 provides many new features, including more than 25 new system procedures. In addition, the catalog stored procedures (formerly installed separately) are now included in the default installation procedures.

Because the new system procedures and system tables nearly double the size of the *master* database, release 10.0 stores system procedures in a new database, *sybsystemprocs*. This saves customers the trouble of repartitioning disks and moving the *master* database to ensure that the entire database resides on a single device.

When your installation or upgrade is complete, Sybase system procedures reside in the new *sybsystemprocs* database. Special search routines for executing any procedure name beginning with "sp_" allow SQL Server to locate the system procedures, even if the procedure call is fully qualified with the database name "master." For example, **master..sp_who** finds the procedure.

Benefits of Moving System Procedures

The new structure provides several benefits:

- It dramatically shrinks the space used in *master*. You will not have to change the size of the *master* database for a long time.
- It creates greater flexibility for users. Users can store larger numbers of local stored procedures in *sybsystemprocs* without worrying about space considerations on the master device.
- It can speed recovery of the *master* database. Under the pre-10.0 structure, if the master device was damaged, a significant portion of recovery time involved the re-creation of the system procedures. Now, if the system procedures are on a separate database device, *master* can be recovered much more quickly.
- The new *sybsystemprocs* database can be restored quickly and easily in case of disk failure without affecting the *master* database.
- The *sybsystemprocs* database can safely use an operating system file, if necessary, as a database device. It is rarely updated and can be easily restored.
- It reduces the size of backups of the *master* database. Due to system restrictions, backups of *master* must fit on a single tape or file.

The Upgrade Process

When you upgrade an existing SQL Server or install a new SQL Server, you specify the device where you want to install *sybsystemprocs*. The installation program creates the device. For upgrades, it drops all existing Sybase system procedures from *master* without affecting any user-created procedures stored in *master*.

If necessary, the upgrade process increases the number of open databases allowed on your SQL Server.

See "Changes That May Affect Existing Applications" on page 4-34 in this manual for information on moving your local procedures from *master* to *sybsystemprocs*, and other system administration information relating to this change.

Backup Server and Space Management

An introduction to Backup Server and associated space management changes follows.

Backup Server

The Open Server™-based Backup Server utility handles all dumps and loads for SQL Server. Dumps and loads now work more quickly and with significantly less effect on other SQL Server activities. New dump and load syntax provides more flexibility and more options:

- Dump striping allows you to use up to 32 dump devices in parallel to dump or load a single database or transaction log.
- Multifile and multivolume dumps allow one dump to span multiple tapes or allow multiple dumps to be made to a single tape.
- Network dumps allow dumping and loading over the network to or from a device on another machine.
- Backup Server complies with ANSI standard tape labeling and handling.
- Platform-specific tape handling options support dump and load command syntax specification for volume naming, dismount and load control, tape density, block size, tape capacity, days to retain, initialization, file naming for multidump volumes, and listing header or file information instead of loading the files.
- The system procedure **sp_volchanged** signals volume changes during backups. (OpenVMS users do this with the **REPLY** command.) It replaces the **console** utility program. **sp_volchanged** can be submitted from any ordinary Sybase client to signal volume changes to the Backup Server.
- Multiple dumps and loads can be managed from one or more local or remote servers.

Installing a Backup Server

If you do not want to use the new functionality immediately, you can continue to use your existing dump and load routines. Pre-10.0 syntax is compatible with 10.0 and later release syntax. The only exception is that **dump database** and **dump transaction** commands cannot be included in a user-defined transaction in release 10.0 and later.

Dump and load commands **must** use Backup Server. It must be installed and running, and SQL Server must be configured to connect to Backup Server.

◆ ***WARNING!***

You cannot load dumps made with pre-10.0 releases using 10.0 dump and load commands: pre-10.0 dumps are not compatible with Adaptive Server 11.5. You can load 10.0 and later dumps using the 11.5 Backup Server.

The installation and configuration guides for your platform contains information about installing Backup Servers, supported dump devices, and other platform-specific issues.

Additional Sources of Information

The new syntax for the **dump database**, **load database**, **dump transaction**, and **load transaction** commands and the **sp_volchanged** system procedure is documented in the *Reference Manual*.

Information about starting and stopping Backup Servers during routine operations can be found under **startserver** in the SQL Server utility programs manual, under **shutdown** in the *Reference Manual*, and in your installation and configuration guide. Also, see **backupserver** in the utility programs manual for your platform for information about flags for the **backupserver** executable.

See the *System Administration Guide*, for a complete discussion of Backup Server.

Threshold Manager

Thresholds monitor how much free space remains on a particular segment. When the amount of free space falls below a threshold, SQL Server automatically executes the associated stored procedure. For example:

- You can create a threshold and a stored procedure to dump the transaction log when space runs low.
- You can create a threshold and a stored procedure that detects when space for the data segment of a database runs low so that messages in the error log warn you before space runs out.

The system procedures **sp_addthreshold**, **sp_dropthreshold**, **sp_modifythreshold,** and **sp_helpthreshold** create, drop, change, and monitor free-space thresholds in a database.

In databases that store data and logs on separate segments, SQL Server installs a last-chance threshold on the log segment. This threshold calls a procedure named **sp_thresholdaction**. Or you can use **sp_modifythreshold** to call a different stored procedure.

Since user requirements for threshold procedures vary so widely, Sybase does **not** provide **sp_thresholdaction**; you must write it yourself. A special system function, **lct_admin**, creates last-chance thresholds on databases created in pre-10.0 SQL Server.

The new **abort tran on log full** option for **sp_dboption** allows you to choose whether to suspend or kill processes that attempt to write to the log when there is not enough log space.

See the *Reference Manual* for specific syntax information and the *System Administration Guide* for information on managing free space with thresholds.

Variables Allowed in Dump and Load Commands

As part of the release 10.0 threshold improvements, you can now use parameters or variables in dump and load commands for database and device names. This allows a single stored procedure (such as **sp_thresholdaction**) to dump multiple databases or transaction logs. See **dump database, dump transaction, load database,** and **load transaction** in the *Reference Manual*.

New Security Features

SQL Server release 10.0 provides features targeted at the C2 level of trust. Many of the security features provide new flexibility for SQL Server users who do not need all of the C2 functionality. The major features are:

- New system administration roles

- New user identification and authentication features, including password encryption
- A complete system for auditing activity on SQL Server

System Administration Roles

In previous releases, SQL Server allowed only one login, "sa", to perform most system administration tasks. SQL Server release 10.0 recognizes the following security-related operational and administrative roles:

- System Administrator: required for tasks such as manipulating disk space and memory usage, granting and revoking permission to execute **create database** statements, running diagnostic and repair functions that read data pages or recover data or indexes in a controlled manner, granting and revoking the System Administrator role, and modifying, dropping, and locking server login accounts.
- System Security Officer: required for security-related tasks such as creating server login accounts, changing passwords, granting and revoking roles, configuring the password expiration interval, and managing the audit system.
- Operator: used for backing up and restoring databases.

These roles can be granted to and revoked from individual login accounts in SQL Server. A login account can have more than one role. Roles can be granted permissions on objects and commands, similar to groups. Details on roles can be found in the *Security Administration Guide.*

When SQL Server release 10.0 is installed, it still has the default "sa" account, which has the System Administrator, System Security Officer, and Operator roles enabled. For greater accountability for the highly privileged users in your system, it is recommended that you create individual login accounts for users who are to be granted these privileges, grant them their roles, and then lock the "sa" account. If you have automated scripts or programs that log into SQL Server as "sa", see the *Security Administration Guide* for more information.

User Identification

SQL Server uses a variety of mechanisms to positively identify an individual user and to enforce accountability and security:

- Login account locking makes it possible to lock a user's account without dropping the user and the user's objects from all databases.
- A minimum password length of 6 bytes makes passwords more secure. (This does not affect existing user passwords, but all new or changed passwords must be at least 6 bytes.)
- Passwords are stored in *master..syslogins* in encrypted form.
- You can enable optional system-wide password expiration.
- You can enable optional client-side password encryption.
- Recovery mechanisms are provided in the case of lost or expired passwords.

Auditing

➤ ***Note***

11.5 Note: Release 11.5 auditing provides multiple audit tables and other significant changes. See "Auditing Enhancements" on page 1-3.

SQL Server can be configured to provide an audit trail for events such as:

- Server logins and logouts
- Use of any commands that require a special role
- Use of commands that reference a specified object or database
- Deletion of objects
- Execution of stored procedures and triggers
- Any actions performed by a specified user

The audit system includes a new database, *sybsecurity*, which contains the audit trail in a system table called *sysaudits*. Auditing is described in the *Security Administration Guide*.

➤ ***Note***

11.5 Note: How to install auditing is explained in the Adaptive Server Configuration Guide for your platform.

Cursors

SQL Server release 10.0 provides full support for cursors. The following cursor commands are documented in the *Reference Manual*:

declare cursor
fetch
open
close
deallocate

New keywords that allow updating and deleting at cursor positions are described on the **delete** and **update** pages. A new **set** option controls the number of rows returned by a **fetch** command. A new system procedure, **sp_cursorinfo**, provides information about the active cursors in a database.

The *Transact-SQL User's Guide* provides examples of cursor use, including cursors in stored procedures.

Data Definition Enhancements

Enhancements are as follows.

Integrity Constraints

In addition to the rules, triggers, defaults, and unique indexes already provided by SQL Server, you can now specify integrity constraints in the **create table** statement. These include:

- Unique and primary key constraints to ensure unique values in a column
- Referential integrity constraints to guarantee that a primary key exists in another table when you are inserting or updating a foreign key table
- Check constraints to limit the values that can be inserted into a table
- Defaults to specify default values for a column

Constraints can be changed or dropped with **alter table**. You can get information about the constraints defined for a table using the new **sp_helpconstraint** system procedure.

See **sp_helpconstraint**, **create table** and **alter table** in the *Reference Manual* for additional information.

The *Transact-SQL User's Guide* describes how to use the constraints and provides additional examples.

Changes to Views

The **distinct** keyword can now be used in view definition, allowing you to create views that do not contain duplicate rows.

The **with check option** clause has been added to the **create view** statement. When a view is created **with check option**, all rows that are inserted or updated through the view must meet the view criteria.

create view statements can be included in batches with other Transact-SQL statements.

See **create view** in the *Reference Manual* and the *Transact-SQL User's Guide* for more information and examples.

Schemas

The new **create schema** command allows you to create several objects and to grant permissions on those objects in a single statement, which can be committed or rolled back as a unit. See **create schema** in the *Reference Manual*.

Datatypes

Release 10.0 of SQL Server provides these new datatypes:

- *numeric* and *decimal* (*dec*) specify exact numeric values, with specified precision and scale to indicate the total number of digits and the number of digits to the right of the decimal point.
- *double precision* specifies an approximate numeric type.

The following changes have been made to existing datatypes:

- *float* now accepts an optional precision specification.
- The default length for the *char*, *varchar*, *nchar*, and *nvarchar* columns is 1 in **create table** statements, variable declarations, and parameter specifications.

All system datatype names are case-insensitive. SQL Server treats numeric constants in queries (such as **select 2* *colvalue***) as exact numeric types unless they are entered using E notation. Values between $2^{31} - 1$ and -2^{31} without decimal points are treated as *int*. Values that fall outside the range for integers, or that include a decimal point, are treated as *numeric*.

See “Changes That May Affect Existing Applications” on page 4-34 for information on how this can affect existing applications.

The following list shows synonymous datatype names. Names that are new in 10.0 are in bold type.

- *char, character*
- *varchar, char varying, character varying*
- *nchar, national character, national char*
- *nvarchar, national character varying, national char varying, nchar varying*
- ***double precision***, *float*, ***real***
- ***dec***, ***decimal***, ***numeric***
- *int, integer*

See the *Transact-SQL User’s Guide* for more information on datatypes.

Automatic Sequential Values Using the IDENTITY Property

Each table in a database can have a single IDENTITY column. This column is used to store numbers, such as invoice numbers or employee numbers, that are automatically generated by SQL Server. The value of the IDENTITY column uniquely identifies each row in a table.

The IDENTITY column must be of type *numeric* and scale 0. By default, IDENTITY columns cannot be updated and do not allow null values. Only the table owner, the Database Owner, or a System Administrator can explicitly insert a value into an IDENTITY column after setting **identity_insert on** for the table.

Each time you insert a row into a table, SQL Server assigns a unique, sequential value to the IDENTITY column. To retrieve the last value inserted into an IDENTITY column, use the *@@identity* global variable. To select a table’s IDENTITY column, use the **syb_identity** keyword, qualified by the table name where necessary.

For more information about IDENTITY columns, see **create table**, **alter table**, **insert**, **select**, and **create view** in the *Reference Manual*. Also see the *Transact-SQL User’s Guide.*

Automatic Creation of IDENTITY Columns

System Administrators can use the new **auto identity** database option to automatically include a 10-digit IDENTITY column in new tables.

Each time a user creates a table in the database without specifying either a **primary** key, a **unique** index, or an IDENTITY column, SQL Server automatically defines an IDENTITY column for the table.

For more information, see the *Transact-SQL User's Guide.*

Changes to Transactions, Triggers, and Stored Procedures

Changes are as follows.

Data Definition Language in Transactions

Certain data definition language commands are now allowed in user-defined transactions. These commands include:

alter table	**create table**	**drop procedure**
create default	**create trigger**	**drop rule**
create index	**create view**	**drop table**
create procedure	**drop default**	**drop trigger**
create rule	**drop index**	**drop view**
create schema		

grant and **revoke** commands are now allowed in transactions. However, some commands such as **dump database** and **dump transaction** are no longer allowed.

A new option for **sp_dboption**, called **ddl in tran**, allows enabling or disabling the use of data definition language in transactions. Be sure to read the cautionary statements about using data definition language in transactions, since this activity requires locking system tables.

For information about which commands you can use in transactions and about the **ddl in tran** option, see the *Transact-SQL User's Guide.*

Transaction State Information: @@*transtate*

SQL Server can now return information about the state of a transaction. This status information indicates whether the transaction succeeded, whether a single statement was rolled back, or whether an entire transaction was rolled back. The information is available in the new global variable, *@@transtate*.

For more information on transaction and global variables, see the *Transact-SQL User's Guide* .

Also see "Variables" in the *Reference Manual* for a list of the status values.

New *rollback trigger* Command

The rollback feature has been expanded to include **rollback trigger.** Now, you can roll back an entire transaction using **rollback transaction**, or roll back statements up to the first data modification that fired a trigger using **rollback trigger.**

See **rollback trigger** in the *Reference Manual* for more information. See also Chapter 18, "Transactions: Maintaining Data Consistency and Recovery," in the *Transact-SQL User's Guide.*

Trigger Self-Recursion

A new **set** option, **self_recursion**, allows triggers to fire self-recursively; that is, to refire as a result of data modifications made by the trigger itself. By default, when a trigger causes data modifications that would also cause the trigger to fire, the trigger is prevented from firing. See **set** in the *Reference Manual* for more information.

Stored Procedure Size Limits Removed

Release 10.0 eliminates the size limit for compiled stored procedures and batches. To accommodate this change, the maximum amount of text allowed for a stored procedure has been increased to 16MB. This increase also applies to objects whose text is stored in the *syscomments* system table, such as triggers, views, defaults, rules, and constraints.

Changes to Datatype Conversions and Queries

Changes are as follows.

Datatype Conversion Changes

Changes were made to datatype conversion rules for release 10.0 to provide greater consistency across all Sybase products.

- Formerly disallowed conversions to and from binary datatypes are now allowed.
- Conversion from integer to character now returns an error instead of "**" if the character value is not long enough to store the converted value.
- Some changes were made to the "datatype hierarchy" that controls the output datatype of mixed-mode datatype operations.

See "Changes That May Affect Existing Applications" on page 4-34 for information on how these changes may affect existing applications.

New Hex Conversion Functions

Two new functions provide platform-independent conversions between integer values and hexadecimal strings.

See Chapter 6, "Using and Creating Datatypes," in the *Transact-SQL User's Guide* for information about **inttohex** and **hextoint**.

Subquery Changes

The behavior of certain subqueries has changed.

See "Changes That May Affect Existing Applications" on page 4-34 for detailed information about these changes and how they may affect your existing applications.

Grouping on *bit* Columns Allowed

You can now use **group by** on *bit* columns.

Random Number Generator Improvements

The **rand** function now uses the output of a 32-bit pseudo-random integer generator.

See Chapter 10, "Using the Built-In Functions in Queries," in the *Transact-SQL User's Guide*.

between Predicate Changes

In some cases, the **between** predicate in Transact-SQL returned results regardless of whether the values supplied were in the proper order

(lower number first after the predicate) or were swapped (higher number first). In release 10.0, a query with swapped **between** values returns no rows.

select into Column Headings

In a **select into** statement, column headings must be provided for any select list item that includes aggregate functions or expressions.

See "Changes That May Affect Existing Applications" on page 4-34 for examples of expressions that require headings and for examples and syntax.

Column Alias

You can now use the keyword **as** in **select** statements between the select expression and the alias name. For example:

```
select au_lname as lastname from authors
```

Full Dynamic SQL/Precompiler Support

SQL Server release 10.0 provides full support for host variables and Dynamic SQL. For complete information, see the *Embedded SQL/C* and *Embedded SQL/COBOL Programmer's Guides.*

Right Truncation of Character Strings

In previous releases, SQL Server silently truncated *char, nchar, varchar*, and *nvarchar* strings when an **insert** or **update** entered strings that were longer than the specified column length. A new **set** option, **string_rtruncation**, controls silent truncation of character strings. Set this option **on** to prohibit silent truncation and enforce SQL92 behavior. See **set** in the *Reference Manual* for more information.

SQL Standards Compliance

In addition to major features such as declarative integrity constraints and cursors, which are described elsewhere in this chapter, the following changes are included in release 10.0 for compliance to SQL standards. Since certain standard behaviors are not compatible with existing SQL Server applications, Transact-SQL provides **set** options that allow you to toggle these behaviors.

set Commands Required for SQL92 Compliance

Compliant behavior is enabled by default for all Embedded SQL™ precompiler applications. Other applications that need to match standard behavior can use the option settings in Table 4-1 for entry level SQL92 compliance.

Table 4-1: set options for compliance to SQL standards

Option	Setting
ansi_permissions	on
ansinull	on
arithabort	off
arithabort numeric_truncation	on
arithignore	off
chained	on
close on endtran	on
fipsflagger	on
quoted_identifier	on
string_rtruncation	on
transaction isolation level	3

The following sections describe the differences between standard behavior and the default Transact-SQL behavior. For more information on setting these options, see **set** in the *Reference Manual.*

FIPS Flagger

For customers who write applications that must conform to standard SQL, SQL Server provides a **set fipsflagger** option to flag incompatible syntax. When this option is turned on, all commands containing Transact-SQL extensions that are not allowed in entry level SQL92 generate an informational message. See **set** in the *Reference Manual* for more information.

SQLSTATE Messages and Codes

By default, SQL Server returns SQLSTATE values to embedded SQL applications, as required by entry level SQL92. SQLSTATE values are

included in a new column in *sysmessages*, when SQLSTATE codes are defined in the standard. Some of the **set** commands listed in Table 4-1 on page 4-16 cause the associated message text to be delivered to client applications such as **isql**.

The *escape* Clause in the *like* Predicate

SQL Server now supports the SQL standard's **escape** clause, which allows you to specify an escape character in the **like** predicate to search for wildcard characters. This is in addition to Transact-SQL's use of square brackets for the same purpose.

See the *Transact-SQL User's Guide* for more information on wildcard characters.

Standard-Style Comments

In Transact-SQL, comments are delimited by /* and */ and can be nested. Transact-SQL now also supports SQL standards-style comments, which consist of any string beginning with two consecutive minus signs, a comment, and a terminating newline character:

```
select "hello" -- this is a comment
```

This syntax conflicts with the subtraction of a negative number.

See "Changes That May Affect Existing Applications" on page 4-34 for information on how this can affect existing applications. Transact-SQL's /**/ comments are still fully supported, and the "--" within Transact-SQL comments is not recognized.

Changes to *set* Options *arithabort* and *arithignore*

Changes were made to the **set** options **arithabort** and **arithignore** to allow compliance with the SQL92 standard:

- **arithabort arith_overflow** determines how SQL Server handles arithmetic overflows and divide-by-zero errors during implicit and explicit conversions. Set this option **on** to roll back the entire transaction or batch in which the error occurs. Set this option **off** to abort the statement that contains the error and continue processing other statements in the transaction or batch.
- **arithabort numeric_truncation** determines how SQL Server handles loss of scale during implicit conversions to a *numeric* or *decimal* datatype. Set this option **on** to abort the statement that contains

the error and continue processing other statements in the transaction or batch. Set it **off** to truncate the results as necessary and continue processing.

- **arithignore arith_overflow** determines whether SQL Server ignores arithmetic overflow and divide-by-zero errors. Set this option **off** to print a warning message when these errors occur. Set it **on** to ignore these errors and print no messages.

If you have used these options in your applications, examine them to be sure that they are still producing the desired behavior.

Synonymous Keywords

Several keywords have been added for SQL standard compatibility that are synonymous with existing Transact-SQL keywords.

Table 4-2: SQL standard-compatible keyword synonyms

Pre-Release 10.0 Transact SQL	Additional Syntax
tran **transaction**	**work**
any	**some**
grant all	**grant all privileges**
revoke all	**revoke all privileges**
max (*expression*)	**max** ([all | distinct]) *expression*
min (*expression*)	**min** ([all | distinct]) *expression*
user_name (built-in function)	**user** keyword

The new **work** keyword is synonymous with **tran** and **transaction** only with the **commit transaction** and **rollback transaction** commands, not with the **begin transaction** command.

See “Datatypes” on page 4-10 for a list of synonymous datatypes.

Chained Transactions and Isolation Levels

SQL Server now provides SQL standard-compliant **chained** transaction behavior as an option. In chained mode, all data retrieval and modification commands (**delete**, **insert**, **open**, **fetch**, **select**, and **update**) implicitly begin a transaction. Since such behavior is incompatible with many Transact-SQL applications, Transact-SQL style (or **unchained**) transactions remain the default.

◆ ***WARNING!***

Before you change the default transaction mode or isolation level, be sure to read the sections in the *Transact-SQL User's Guide* and the *Reference Manual* that describe how these changes can affect existing applications and stored procedures.

Chained transaction mode can be initiated with a new option to the **set** command. Another **set** option controls transaction isolation levels.

See Chapter 18, "Transactions: Maintaining Data Consistency and Recovery," in the *Transact-SQL User's Guide*.

Delimited Identifiers

SQL Server now supports delimited identifiers for table, view, and column names. Delimited identifiers are object names enclosed in double quotation marks. Using them allows you to avoid certain restrictions on object names.

Delimited identifiers can begin with non-alphabetic characters, including characters that would not otherwise be allowed, or even be Transact-SQL reserved words. They cannot exceed 28 bytes.

Set the new **quoted_identifier** option **on** to allow delimited identifiers. While the option is on, do not use double quotes around character or date strings; use single quotes instead. Delimiting strings with double quotes causes SQL Server to treat them as identifiers.

➤ ***Note***

Delimited identifiers cannot be used with some system procedures, cannot be used with **bcp**, and may not be supported by all client software.

Treatment of Nulls

A new **set** option, **ansinull**, determines whether or not evaluation of NULL-valued operands in SQL equality (=) or inequality (!=) comparisons and in aggregate functions is SQL standard-compliant. This option does not affect how SQL Server evaluates NULL values in other kinds of SQL statements, such as **create table**.

Right Truncation of Character Strings

A new **set** option, **string_rtruncation**, controls silent truncation of character strings for SQL standard compatibility. Set this option **on** to prohibit silent truncation and enforce SQL standard behavior.

Permissions Required for *update* and *delete* Statements

A new **set** option, **ansi_permissions**, determines what permissions are required for **delete** and **update** statements. When this option is **on**, SQL Server uses SQL92's more stringent permissions requirements for these statements. Because this behavior is incompatible with many existing applications, the default setting for this option is **off**.

Enhancements to Permissions

Object and command permissions in SQL Server have been enhanced with the following new options:

grant with grant Option

A new clause for the **grant** command, **with grant option**, allows a permission holder to pass **grant** capability, along with the specific permission, to other users, but not to groups, roles, or "public." A corresponding clause for the **revoke** command, **cascade**, allows the privilege holder to revoke permissions from all users who obtained it via **with grant option**.

See **grant** and **revoke** in the *Reference Manual* for syntax information. Also see the *Security Administration Guide* for a full discussion and examples of this new option.

Granting to Roles

You can grant and revoke object and command permissions to all users who have been granted a specific role. The roles are **sa_role**, **sso_role**, and **oper_role**. Permissions granted to roles override permissions granted to individuals or groups.

Configurable Packet Size

New **sp_configure** options allow System Administrators to reset the network packet size used by SQL Server. Larger packet sizes can

yield performance improvements when large amounts of data are being transferred into or out of SQL Server.

See the *System Administration Guide* for full information.

New command line options for **isql** and **bcp** allow you to set the packet size for an individual session. See the SQL Server utility programs manual for information about using these options. Also see the Open Client Client-Library™ documentation for information on using variable packet sizes.

Chargeback Accounting

Chargeback accounting provides a mechanism for tracking CPU and I/O usage for each server user. This feature was previously available only on OpenVMS.

See **sp_clearstats** and **sp_reportstats** in the *Reference Manual* for information about getting reports and restarting the accounting interval. See **sp_configure** for information about the **cpu accounting flush interval** and **i/o accounting flush interval** configuration parameters that affect chargeback accounting.

New *dbcc* Options

These changes have been made to **dbcc**, the database consistency checker:

- A **fix** option has been added to **dbcc checkalloc**, so this command can now fix some allocation errors.
- A **skip_ncindex** option has been added to **dbcc checkdb** and **dbcc checktable**. When you use this option, **dbcc** skips checking the nonclustered indexes. Depending on the number of indexes on your table or in your database, this can speed up the performance of the command by up to 40 percent.

kill Command Enhancements

The **kill** command, used to terminate SQL Server sessions, can now terminate more kinds of sessions. In previous releases, some of these sessions could not be terminated (except by rebooting SQL Server), or would be terminated only when the processes stopped sleeping. Now, these sessions can be terminated immediately. The **sp_who** system procedure displays different status values for these sessions.

For more information, see **kill** in the *Reference Manual* and in the *System Administration Guide.*

shutdown Command Enhancements

The **shutdown** command, used to shut down a SQL Server, can now be used to shut down a Backup Server. The Backup Server must be listed in your *sysservers* table and in the *interfaces* file for the SQL Server from which you execute the **shutdown** command.

Unless you use the **with nowait** option, **shutdown** waits for active dumps and loads to complete. Once you issue a **shutdown** command to a Backup Server, no new dumps or loads to the Backup Server can start.

tempdb Changes

By default, all users now have **create table** permission in *tempdb*.

See the *Transact-SQL User's Guide* for more information on temporary tables.

Additional Information on Space Usage

System procedures now display additional information about space available in databases:

- **sp_helpdb *database_name*** displays information about how much space is left on each disk piece assigned to the database
- **sp_helpsegment *segment_name*** displays information about the amount of free space left on the segment

Error Handling

raiserror can now return the names of the tables and columns that are causing an error, along with the error number and text. See **raiserror** in the *Reference Manual*.

create index Performance Enhancements

➤ ***Note***

11.5 Note: This section is superceded by release 11.5. Skip this section.

Internal improvements have enhanced the speed of the **create index** command. Also, you can increase the speed of creating indexes with a new **sp_configure** option.

When SQL Server creates an index, it reads and writes pages to disk for intermediate sort results and writes out the final index pages one page at a time. With release 10.0, a System Administrator can allocate buffers so that **create index** can perform these reads and writes one extent at a time, thereby making more efficient use of the disk and increasing the speed of the **create index** command. If you want to use this option, there are two important considerations:

- The buffers are **added to** the memory allocated by SQL Server. Setting the value too large can make it impossible for SQL Server to start if it cannot acquire enough memory.
- The first user who starts a **create index** command acquires all of the allocated buffers. All subsequent **create index** commands that are started before the first one completes will use regular one-page-at-a-time disk I/O. For best performance, either create indexes when other users are not likely to contend for the resource or coordinate the creation of large indexes among table owners.

See Chapter 11, "Setting Configuration Parameters," of the *System Administration Guide* for information about the new **sp_configure** option, **number of extent i/o buffers.** The section contains suggestions for choosing the correct buffer size.

Improvements to the Query Optimizer

➤ ***Note***

11.5 Note: You should run **update statistics** after upgrading to 11.5.

Improvements to the query optimizer have increased SQL Server's accuracy in evaluating the cost of using indexes. You must run **update statistics** on your indexes in order to take advantage of these changes, although any previously created indexes continue to work.

This change involves the way that the query optimizer assesses the cost of using indexes with compound keys on the inner table of a join. Prior to release 10.0, the optimizer computed a value for the entire key and often underestimated the cost of using an index if the query used only one or a few of the columns. In release 10.0, SQL Server maintains statistics for each prefix of columns in the index, that is, for the first column, for the first and second columns, for the first, second, and third columns, and so on.

The statistics used by the optimizer are generated by the **create index** command and are updated by the **update statistics** command. Indexes created in earlier releases of SQL Server still work in release 10.0, but in order to take advantage of the new behavior described here, you must run the **update statistics** command on the table or index.

Bulk Copy Performance Enhancements

➤ ***Note***

11.5 Note: Release 11.5 adds the ability to use parallel BCP with partitioned tables. See “Parallel Bulk Copy” on page 1-22.

Internal improvements, which are transparent to users, result in faster bulk copy speeds. In addition, configurable packet sizes can increase bulk copy throughput.

See **bcp** in the utility programs manual for your platform and Chapter 11, “Setting Configuration Parameters,” in the *System Administration Guide* for information about configuring packet size.

Spanish Collating Orders

Release 10.0 supports these Spanish dictionary sort orders:

- Case-sensitive
- Case-insensitive
- Case- and accent-insensitive

To change the sort order on an existing SQL Server to use a Spanish collating sequence, follow the directions in Chapter 13, “Configuring Character Sets, Sort Orders, and Languages,” in the *System Administration Guide.*

Instructions for installing a new SQL Server using a Spanish collating sequence are in the installation and configuration guides for your platform.

System Changes in Release 10.0

This section provides a summary of release 10.0 changes. Those changes are:

set Options

The following table summarizes new and changed **set** options:

Table 4-3: New set options

Option	Function
set ansinull {on | off}	Toggles SQL standard treatment of NULL-valued operands in equality (=) and inequality (!=) comparisons. When **on**, treatment is SQL92-compliant.
set ansi_permissions {on | off}	Determines whether SQL92-compliant permissions requirements for **update** and **delete** statements are checked.
set arithabort [arith_overflow | numeric_truncation] {on | off}	Determines whether SQL Server aborts a query when an arithmetic overflow or numeric truncation error occurs.
set arithignore [arith_overflow] {on | off}	Determines whether SQL Server displays a warning message after any query that results in arithmetic overflow.
set chained {on | off}	Toggles the chained transaction mode in SQL Server.

Table 4-3: New set options (continued)

Option	Function
set close on endtran {on | off}	Forces all open cursors to be closed when the transaction ends (**on**) or to remain open across transactions (**off**).
set cursor rows *number* for *cursor_name*	Sets the number of cursor rows returned to the host on each **fetch** for the specified client cursor. The default is 1.
set dup_in_subquery {on | off}	Controls whether a subquery using the **in** clause returns duplicate values. The default is **off**; duplicate rows are not returned. In prior SQL Server releases, a row was returned for each matching row in the subquery. SQL92 specifies removing duplicate values from the result set.
set fipsflagger {on | off}	Toggles the FIPS flagger. When **on**, the option prints warning messages for the Transact-SQL extensions to SQL92.
set identity_insert *table_name* {on | off}	Determines whether a user can explicitly insert a value into *table_name*'s IDENTITY column. Users can turn this option **on** for only one table at a time per database.
set quoted_identifier {on | off}	Determines whether SQL Server treats strings enclosed in double quotes (") as identifiers.
set role {"sa_role" |"sso_role" | "oper_role"} {on | off}	Toggles the specified role during the current SQL Server session.
set self_recursion {on | off}	Switches trigger self-recursion on or off so that triggers that modify data cause the trigger to fire again.
set string_rtruncation {on | off}	Determines whether silent truncation of character strings is allowed. When **on**, treatment is SQL92-compliant.
set transaction isolation level {1|3}	Sets the transaction isolation level. An isolation level of 3 is the equivalent of **select with holdlock**.

set showplan now prints full command names instead of abbreviations.

New and Changed Built-In Functions

The following table lists new and changed built-in functions:

Table 4-4: New and changed built-in functions

Name	Function
col_name	Changed to accept a database name as an optional third parameter.
curunreservedpgs	New system function. Returns the number of free pages in a disk piece.
inttohex	New datatype conversion function. Returns the platform-independent hexadecimal equivalent of an integer.
hextoint	New datatype conversion function. Returns the platform-independent integer equivalent of a hexadecimal string.
index_col	Changed to accept a user name as an optional fourth parameter.
lct_admin	New system function. Adds a last-chance threshold on the log segment of a pre-10.0 database, or reports on the last-chance threshold status, or wakes up processes waiting for a threshold, depending on the parameter you use.
object_name	Changed to accept a database name as an optional second parameter.
proc_role	New system function. Checks to see if the user possesses the specified role. Useful for restricting execution of procedures.
show_role	New system function. Shows the user's current, active roles.
user	New SQL92-compatible system function returns the user name.

In addition, many of the mathematical functions have been changed to accept the new numeric datatypes, as appropriate.

See the *Reference Manual* for information on these functions.

New System Procedures

The following new system procedures provide some of the functionality for the features described earlier in this summary. The table shows the procedure name and its associated function:

Table 4-5: New system procedures

Procedure	Function
sp_addauditrecord	Allows users to enter user-defined audit records (comments) into the audit trail.
sp_addthreshold	Creates a free-space threshold to monitor space remaining on a database segment.
sp_auditdatabase	**Note:** This procedure is replaced by **sp_audit** in release 11.5. Establishes auditing of different types of events within a database or of references to objects within that database from another database.
sp_auditlogin	**Note:** This procedure is replaced by **sp_audit** in release 11.5. Audits a user's attempts to access tables and views, and/or the text of a user's commands.
sp_auditobject	**Note:** This procedure is replaced by **sp_audit** in release 11.5. Establishes auditing of accesses to tables and views.
sp_auditoption	**Note:** This procedure is replaced by **sp_audit** in release 11.5. Enables and disables system-wide auditing and global audit options.
sp_auditsproc	**Note:** This procedure is replaced by **sp_audit** in release 11.5. Audits the execution of stored procedures and triggers.
sp_bindmsg	Binds a user message to an integrity constraint.
sp_checkreswords	Checks for reserved words used as identifiers. The procedure is run as part of pre-upgrade.
sp_configurelogin	Initializes the security-relevant information for a new Secure SQL Server™ login.
sp_cursorinfo	Reports information about a specific cursor or all cursors that are active.

Table 4-5: New system procedures (continued)

Procedure	Function
sp_dbremap	Forces changes made by **alter database** to be recognized by SQL Server. Run this procedure only if instructed by SQL Server messages.
sp_displaylogin	Displays information about a login account.
sp_dropthreshold	Removes a free-space threshold from a segment.
sp_estspace	Estimates the amount of space needed for a table and its indexes.
sp_helpconstraint	Reports information about any integrity constraints specified for a table.
sp_helpthreshold	Reports information about all thresholds in the current database or all thresholds for a particular segment.
sp_locklogin	Prevents a user from logging in, or displays a list of all locked accounts.
sp_modifylogin	Modifies default database, default language, and full name information for a SQL Server™ login account.
sp_modifythreshold	Changes parameters for existing thresholds in a database.
sp_procxmode	Displays or changes the transaction modes associated with stored procedures.
sp_remap	Upgrades a release 4.8 or higher stored procedure, trigger, rule, default, or view to be compatible with release 10.0. Use this on objects that the **upgrade** procedure failed to remap.
sp_role	Grants or revokes roles to a SQL Server login account.
sp_thresholdaction	Default threshold procedure that is executed automatically when remaining space on the log segment falls below the last-chance threshold. This procedure is not supplied by Sybase. By creating it yourself, you ensure that it is tailored to your own needs.
sp_unbindmsg	Unbinds a user-defined message from a constraint.
sp_volchanged	Notifies SQL Server that the operator has performed the requested volume handling during a **dump** or **load**.

The following procedures are used for chargeback accounting. The chargeback accounting feature is now available on all platforms. In earlier releases, it was available only on OpenVMS.

Table 4-6: Chargeback accounting system procedures

Procedure	Function
sp_clearstats	Initiates a new accounting period for all server users, or a specified user. Prints out statistics for the previous period by executing **sp_reportstats**.
sp_reportstats	Reports statistics on system usage.

See the *Reference Manual* for information on these procedures.

Changes to System Procedures

The following system procedures changed in release 10:

Table 4-7: Changes to system procedures

Procedure	Change
sp_addsegment **sp_dropsegment** **sp_extendsegment**	These procedures now require a database name as the second argument. This change prevents users from accidentally affecting segments in the wrong databases.
sp_defaultdb **sp_defaultlanguage**	These procedures have been superseded by **sp_modifylogin**, which can also change or add "full name" information about a login account. These procedures are still provided by Sybase, but are no longer documented. Use **sp_modifylogin** instead.
sp_helpdb	Now displays information about how much space is left on each disk piece assigned to the database.
sp_helpsegment	Now displays information about the amount of free space left on the segment.
sp_lock	Now displays information whether a lock is associated with a cursor.
sp_who	Displays different status values for sleeping processes: recv sleep, send sleep, lock sleep, and alarm sleep.

New *sp_configure* Configuration Parameters

➤ ***Note***

11.5 Note: Extensive changes were made to the naming of configuration parameters in release 11.0. See “New Features in Release 11.0” on page 3-1 as well as “New Features in Release 11.5” on page 1-1.

The following are new **sp_configure** parameters added in release 10.0:

Table 4-8: New configuration parameters

Option	Function
additional netmem	Memory added for use with variable packet sizes.
audit queue size	Number of audit records in the audit queue.
cpu accounting flush interval (formerly **cpu flush**)	Used to configure chargeback accounting (new on non-OpenVMS platforms).
default network packet size	Default packet size.
identity burning set factor	Determines the percentage of potential IDENTITY column values made available in memory. When assigning a new IDENTITY column value, SQL Server chooses the next value from this set. If the server fails before assigning these values, it discards any remaining values in the set, leading to gaps in IDENTITY column values.
i/o accounting flush interval (formerly **i/o flush**)	Used to configure chargeback accounting (new on non-OpenVMS platforms).
max network packet size (formerly **maximum network packet size**)	Maximum allowable packet size.
number of extent i/o buffers (formerly **extent i/o buffers**)	**Note:** This parameter was dropped in release 11.5. Allocates additional memory to be used for buffering data pages during **create index** and **order by** reads and writes to disk.
systemwide password expiration (formerly **password expiration interval**)	Sets password expiration time.

The serial number field, configuration value 114, has been removed.

New Database Options

The following are new database options added in release 10.0:

Table 4-9: New sp_dboption database options

Option	Function
abort tran on log full	Determines whether user processes are suspended (the default) or aborted when the last-chance threshold on a database's log segment is reached.
allow nulls by default	Changes the default null type for **create table** statements from **not null** (the Transact-SQL default) to **null** (the SQL92 default.)
auto identity	Automatically defines a 10-digit IDENTITY column, **syb_identity_col**, in each new table that does not specify a **primary** key, a **unique** index, or an IDENTITY column. The column is not visible with a **select *** statement; to retrieve it, you must include the column name in the select list.
ddl in tran	Allows data definition language in user transactions. The allowed commands are all **create** and **drop** commands, except **create/drop database**, plus **grant** and **revoke** commands.
no free space acctg	Suppresses free-space accounting and the execution of threshold actions on non-log segments of a database.

You can change these options with **sp_dboption.**

Also, the **trunc. log on chkpt.** option can now be used with or without the periods, that is, **trunc log on chkpt** is also supported.

New System Tables

Four new system tables for all databases were introduced with release 10.0:

- *sysconstraints*
- *sysreferences*
- *sysroles*
- *systhresholds*

The *master* database includes the following new system tables:

- *syslisteners*
- *sysloginroles*

- *syssrvroles*

If you install auditing, the *sybsecurity* database is automatically created with two tables: *sysaudits* and *sysauditoptions* (as well as the default system tables for all user databases).

Changes to Existing System Tables

The following system tables were changed in this release:

Table 4-10: Changes to existing system tables

Table	Change
master..sysdatabases	New columns: *status2, audflags, deftabaud, defvwaud, defpraud; mode* removed.
master..syslocks	New column: *class.*
master..syslogins	New columns: *pwdate, audflags, fullname.* Formerly reserved columns now used: *status, accdate, totcpu,* and *totio.* (The last three columns are used by chargeback accounting, formerly available on OpenVMS only.)
master..sysmessages	New column: *sqlstate.*
master..sysprocesses	New columns: *tran_name, time_blocked, network_pktsz.*
master..sysusages	New columns: *pad and unreservedpgs*.
syscolumns	New columns: *prec, scale.*
syscomments	New column: *colid2.*
sysindexes	New column: *status2.* Changed columns: *spare1* to *oampgtrips, spare2* to *ipgtrips.*
sysobjects	Changed columns: *schema* to *schmacnt, refdate* to *sysstat2, category* to *ckfirst.* New columns: *objspare* and *audflags.*
sysprotects	New column: *grantor.* Changed columns: values for the *protecttype* column are different. 0 indicates **grant with grant**, 1 indicates regular **grant**, and 2 indicates **revoke**.
systypes	New columns: *prec, scale, ident, hierarchy.*

Changes That May Affect Existing Applications

This section describes system changes introduced by release 10.0 that may affect your applications if you are upgrading to release 11.5 from a pre-10.0 release. Those changes are:

New Transact-SQL Keywords for Release 10.0

The following keywords are new “reserved” words in SQL Server 10.0. They cannot be used as object names or column names.

Table 4-11: New release 10.0 reserved words

arith_overflow	identity_insert	references
at	isolation	replace
authorization	key	role
cascade	level	rows

Table 4-11: New release 10.0 reserved words (continued)

check	mirror	schema
close	national	shared
constraint	noholdlock	some
current	numeric_truncation	stripe
cursor	of	syb_identity
deallocate	only	syb_restree
double	open	user
endtran	option	user_option
escape	precision	varying
fetch	primary	work
foreign	privileges	
identity	read	

You must change all database names that are new reserved words before you can upgrade an earlier version of the server. You can change table, view, and column names or use delimited identifiers. Once you upgrade, you cannot use database objects whose names are new reserved words until you modify your procedures, SQL scripts, and applications.

A new system procedure, **sp_checkreswords**, checks for reserved words used as identifiers and reports the object names. If you choose to leave some of these reserved words as identifiers until after you upgrade, you can run the procedure **sp_checkreswords** at any time. See **sp_checkreswords** in the *Reference Manual* for information about running this procedure and for detailed information about upgrading your applications.

Password Changes

SQL Server no longer allows NULL passwords. All passwords must be at least 6 bytes in length. Existing passwords containing less than 6 bytes are not changed during upgrade, but all future uses of **sp_password** will require a new 6-character password.

You can globally force users to change passwords within a specified time period using the **systemwide password expiration** option to **sp_configure**. This feature provides last-chance warning messages when a user logs in with a password that is about to expire. Users

whose passwords have expired can log in, but they can execute **sp_password** only.

If you have applications that include embedded passwords, you will need to update them each expiration period if you turn on password expiration.

See the *System Administration Guide* for more information.

Addition of *sybsystemprocs* Database

As of release 10.0, all Sybase-provided system procedures are stored in a database called *sybsystemprocs* rather than in the *master* database. Possible effects on user applications are:

- Permissions on system procedures need to be changed in *sybsystemprocs.*
- System Administrators need to decide where to place their own procedures.
- Upgrade may increase the **open databases** configuration parameter, if adding *sybsystemprocs* requires it. You should verify the value for your needs on your server.
- Scripts that perform backups and **dbcc** may need to be changed.
- Recovery procedures of the *master* database and system databases have been changed.

Changing Permissions on System Procedures

If you have changed the default permissions on any system procedures, you must **grant** or **revoke** the permissions on those procedures in *sybsystemprocs* after upgrade.

Moving Your Own Procedures

To move your own procedures to *sybsystemprocs*, you must:

- Change all references to system tables or other objects that exist only in *master* to reference the *master* database explicitly.
- Add checks to be sure that the procedure is not being run from within a transaction.

Once these steps have been performed, you can create the procedure in the *sybsystemprocs* database and drop it from *master.*

Changing References to master's System Tables

If you want to move your locally created system procedures to *sybsystemprocs*, and they reference system tables in *master*, you must qualify all the table names with the database name.

For example, if your procedure references *syslogins*, you must change the reference to *master..syslogins*. When you create a stored procedure, SQL Server checks to see whether all the tables that are referenced in the procedure exist. You cannot create the procedure in *sybsystemprocs* without the explicit database reference.

Checking for the Existence of Transactions

A stored procedure in *sybsystemprocs* should never modify the tables in *master* inside a transaction, because this could create problems with recovery.

Sybase system procedures include a check, similar to this:

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

Configuring Open Databases

By default, SQL Server is configured to allow up to 10 open databases. The upgrade process will reconfigure the **open databases** variable if the addition of *sybsystemprocs* requires an additional open database.

Changing Backup and *dbcc* Procedures

If you add your own procedures to *sybsystemprocs* or change the default permissions on the system procedures, you should add *sybsystemprocs* to your regular backup schedule.

You should also include *sybsystemprocs* in your regular **dbcc** checks.

Changes in Master Recovery

Due to the addition of *sybsystemprocs* and also Backup Server changes in system tables and system procedures, the process of recovering system databases in case of a failure of the master device has changed.

See the *System Administration Guide* for more information.

Remote Access and the Backup Server

As of release 10.0, all dump and load operations are executed through remote procedure calls to the Backup Server. To allow SQL Server to communicate with the Backup Server, the configuration parameter **allow remote access** must be turned on. Leave this parameter turned on unless you have security concerns. You must be a System Security Officer to set **allow remote access**.

Turning **allow remote access** on does not affect any of the other conditions needed to execute remote procedure calls between servers. This setting, in itself, does not represent a security concern, because server-to-server communication requires additional system administration actions to enable remote procedure calls. These actions include making entries for the remote servers in *master..sysservers*; and making entries for remote users in *master..sysremotelogins*.

If you want to leave **allow remote access** disabled except during a dump or load, you must issue the following command before issuing the dump or load command:

```
sp_configure "allow remote access", 1
```

After you complete the dump, turn **allow remote access** off again with the command:

```
sp_configure "allow remote access", 0
```

The **allow remote access** configuration parameter is dynamic: you do not have to restart SQL Server for the **sp_configure** command to take effect.

Changes to Dump Scripts

Backup facilities have been changed as of SQL Server release 10.0. This section describes the minimum changes needed to existing dump scripts.

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.

➤ ***Note***

All dumps to the "null device" (*/dev/null* on UNIX or any device starting with "NL" on OpenVMS) are now prohibited.

Here are some guidelines for backing up your databases immediately after upgrading:

- If you use new tapes, or tapes without ANSI labels, existing dump scripts overwrite the entire tape.
- If you use single-file media (for example, 1/4-inch cartridge) with ANSI labels, and the expiration dates on the tapes have expired, existing 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 overwrites the existing tape; a negative response initiates a request for a volume change, and tests are repeated on the new volume.
- If you use multifile tape media, and you do not change your dump scripts, the dump is 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 databasse mydb
      to datadump1
      with init
  ```

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

The following diagram shows the logic used in tape label checking on a dump command used without the **init** option:

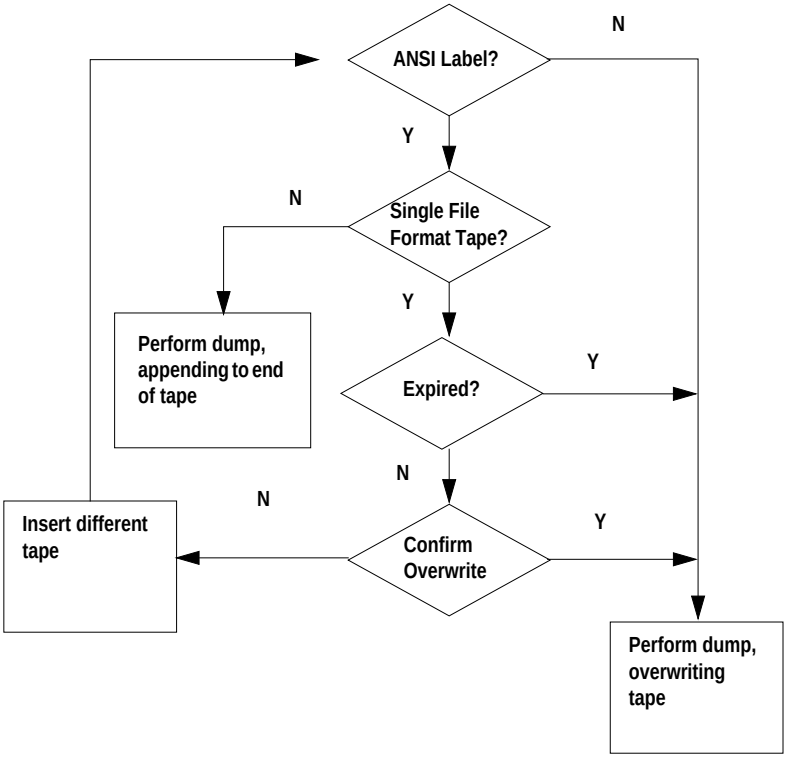

Figure 4-1: Checking tape format and expiration dates for dump commands

Changes to Renaming Databases

If any table in the database references—or is referenced by—a table in another database, **sp_renamedb** cannot rename the database. It produces the following error message:

```
Database 'database_name' has references to other
databases. Drop those references and try again.
```

Execute the following query to determine which tables and external databases have foreign key constraints on primary key tables in the current database:

```
select object_name(tableid), db_name(frgndbname)
from sysreferences
where frgndbname is not null
```

Execute the following query to determine which tables and external databases have primary key constraints for foreign key tables in the current database:

```
select object_name(reftabid), db_name(pmrydbname)
from sysreferences
where pmrydbname is not null
```

Before renaming the database, you must use **alter table** to drop the cross-database constraints in these tables.

See **sp_renamedb** in the *Reference Manual* for more information about renaming databases.

Datatype and Conversion Changes

Changes to datatypes and conversions affect the following areas.

Display Format Changes

The **isql** display format for approximate numeric datatypes now displays additional digits of precision. Maximum units of precision for storage and display are machine dependent. *real* values now display up to 9 digits of precision; *float* values, up to 17 digits. Values are rounded to the last digit on display. Previously, only 6 places to the right of the decimal point 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.

Default Type Changes

Numeric literals in SQL statements are now treated as exact numeric types, *int* or *numeric*, unless they are entered using scientific notation. Here are examples of differences you might see:

Table 4-12: Changes in output using numeric literals

SQL Statement	Pre-10.0 SQL Server	10.0
`select 2147483648`	Integer overflow	2147483648
`select 2*1.12345678`	2.246914	2.24691356
`select 2.8 * 5`	14.000000	14.0

Datatype Hierarchy Changes

Each system datatype has a **datatype hierarchy**, stored in the *systypes* system table. User-defined datatypes inherit the hierarchy of the system types on which they are based. The following query ranks the system datatypes by hierarchy:

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

```
name                           hierarchy
------------------------------ ---------
floatn                                  1
float                                   2
datetimn                                3
datetime                                4
real                                    5
numericn                                6
numeric                                 7
decimaln                                8
decimal                                 9
moneyn                                 10
money                                  11
smallmoney                             12
smalldatetime                          13
intn                                   14
int                                    15
smallint                               16
tinyint                                17
bit                                    18
varchar                                19
sysname                                19
nvarchar                               19
char                                   20
nchar                                  20
varbinary                              21
timestamp                              21
binary                                 22
text                                   23
image                                  24
(28 rows affected)
```

The datatype hierarchy determines the results of computations using values of different datatypes. The result value is assigned the datatype that is closest to the top of the list.

In earlier releases of SQL Server, there were certain exceptions to this rule that have been eliminated as of release 10.0. Specifically, *float* and

numeric computations have been combined with *money* or *smallmoney*.

Table 4-13: Changes to datatype hierarchy

Datatypes	Pre-10.0 SQL Server	10.0
money and *numeric*	N / A	*numeric*
money and *float*	*money*	*float*

Workarounds include:

- If you are combining *money* and literals or variables, and need results of *money* type, use *money* literals or variables:

  ```
  select moneycol * $2.5 from mytable
  ```

- If you are combining *money* with a *float* or *numeric* column, use the **convert** function:

  ```
  select convert (money, moneycol * percentcol)
      from debts, interest
  ```

Overflow Changes

More information on overflow conditions is provided, and overflow behavior has changed for some datatypes. Additional error messages provide more information about the specific cause of the overflow problem.

Floating Point-to-Character Conversions

In previous releases, conversion from floating point to character in some cases allowed some truncation of floating point decimal data without warning, and in other cases generated warning messages.

Now, all conversions to character data of any length succeed only if no decimal digits are lost. The length of the character string varies depending upon the number being converted and the accuracy of floating point numbers supported by the platform. To guarantee success, use a target of 25 characters.

Numeric-to-Numeric Conversions Requiring Truncation

All conversions to money datatypes round to four places. When an explicit conversion of one numeric value to another results in a loss of scale, the results are truncated without warning. For example,

when you explicitly convert a *float*, *numeric*, or *decimal* type to an *integer*, SQL Server assumes you really want the result to be an integer and truncates all numbers to the right of the decimal point.

During implicit conversions to *numeric* or *decimal* types, loss of scale generates a scale error. Use the **arithabort numeric_truncation** option to determine how serious such an error is considered. The default setting, **arithabort numeric_truncation on**, aborts the statement that causes the error but continues to process other statements in the transaction or batch. If you set **arithabort numeric_truncation** to **off**, SQL Server truncates the query results and continues processing.

Integer-to-Character Conversions

Conversions from integer types to character types now return a buffer overflow error (instead of "*") on overflow.

Changes to Conversion Error Messages

The text of the messages for unsupported datatype conversions (message 529) now reads:

```
Explicit conversion from datatype 'type' to 'type'
is not allowed.
```

New error messages provide additional information about datatype conversion errors:

Table 4-14: New type conversion error messages

Error Number	Message
241	Scale error during [explicit | implicit] conversion of *datatype* value '*value*' to a *datatype* field.
245	Domain error during [explicit | implicit] conversion of *datatype* value '*value*' to a *datatype* field.
247	Arithmetic overflow during [explicit | implicit] conversion of *datatype* value '*value*' to a *datatype* field.
249	Syntax error during [explicit | implicit] conversion of *datatype* value '*value*' to a *datatype* field.
265	Insufficient result space for [explicit | implicit] conversion of *datatype* value '*value*' to a *datatype* field.

Change to *between*

The **between** predicate is used in **where** clauses such as this:

```
where colx between val1 and val2
```

val1 must be less than *val2*. In a few situations, the **between** predicate returned correct values, even when *val2* < *val1*. As of release 10.0, these values must be given in the correct order; check your applications that use **between** if you think they may be dependent on the incorrect behavior.

Standard-Style Comments

To comply with SQL92, SQL Server supports standards-style comments, which consist of two minus sign characters. All characters after "--" are ignored. This could cause problems in any SQL query that uses the subtraction of a negative number, such as this:

```
select 5--2
```

This query no longer returns the value 7; it now returns 5. All characters from "--" to the end of the line are ignored.

You can correct any constructions like these by inserting a space between the minus signs or by enclosing the "-2" in parentheses:

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

SQL Standards Permissions Requirements for *update* and *delete*

SQL Server supports SQL92 permissions requirements for **update** and **delete** statements. The following table summarizes these requirements:

Table 4-15: SQL92 permissions for update and delete statements

	Permissions Required with *set ansi_permissions off*	Permissions Required with *set ansi_permissions on*
update	**update** permission on columns where values are being set	**update** permission on columns where values are being set And **select** permission on all columns appearing in **where** clause **select** permission on all columns right side of **set** clause
delete	**delete** permission on columns where values are being set	**delete** permission on the table from which rows are being deleted And **select** permission on all columns appearing in **where** clause

If you attempt to **update** or **delete** without having all the required permissions, an exception is generated and the transaction is rolled back. In this case, you will see the following message:

```
permission_type permission denied on object object_name database
database_name, owner object_owner
```

If this occurs, you need to be granted **select** permission on all relevant columns by the column owner.

SQL Standards Treatment of Nulls

SQL Server supports SQL92-compliant treatment of NULL value operands in equality (=) and inequality (!=) comparisons.

When **set ansinull** is **on**, SQL Server returns an error message if a NULL value operand is eliminated from equality (=) and inequality (!=) comparisons and aggregate functions.

Switching to Chained Transaction Mode

Stored procedures written to use regular Transact-SQL transaction mode may be incompatible with the chained transaction mode required by SQL92. As of release 10.0, all transactions are tagged to identify which mode they use.

If you intend to convert existing SQL Server applications to use chained transaction mode, see **sp_procxmode** in the *Reference Manual*.

Using Roles in SQL Server

➤ ***Note***

11.5 Note: Release 11.5 adds the ability to create user-defined roles in addition to these system roles. See the *Security Administration Guide*.

See "System Administration Roles" on page 4-7 for a description of the new roles. There are two ways to use roles in SQL Server:

- As before, use the "sa" account for system administration tasks. When SQL Server is installed, this account has the System Administrator, System Security Officer, and Operator roles enabled.
- After installing SQL Server, use the "sa" account to create new server logins for users who are to be granted roles. Grant the correct roles to these users, and then lock the "sa" account.

The second method increases user accountability for these highly privileged users, as they perform system administration tasks under their own logins.

However, if you decide to lock the "sa" account, be sure to check all scripts that may contain the "sa" login name and password. These can include scripts that perform backups, run **bcp**, or perform **dbcc** checking. The scripts cannot run if they are meant to run as "sa" and that account is locked. Change the logins in those scripts to the name of one of the users with the correct role.

Repeated Table Names in *from* Clauses

In SQL92, the following construction is not allowed:

```
select * from table1, table1
    [where_clause]
```

In earlier releases, SQL Server handled the **select** statement by returning only the first table and ignoring the self-join. As of release 10.0, this syntax returns an error message.

The correct syntax uses table correlation names:

```
select * from table1 t1, table1 t2
    [where_clause]
```

A similar construction using repeated table names in an **update** statement also returns an error message. The construction is:

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

The **update** statement also creates a self-join on the table named after **update** and the table in the **from** clause. It returns unexpected results because it updates all rows from the first-mentioned table, seemingly ignoring the restriction in the **where** clause. The correct syntax also uses table correlation names in the **from** clause:

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

Note that the SQL92 standard does not include the use of the **from** clause in an **update** statement.

Column Names Required for *select into* with Expressions

Previous releases of SQL Server allowed NULL column headings in tables created by **select into**. These NULL headings resulted from any use of an expression in the select list: an aggregate function, constant, built-in function, arithmetic expression, concatenation, and so on. As of release 10.0, you must provide a column heading. The heading must be a valid identifier, that is, it must begin with an alphabetic character or an underscore, and contain only letters, numbers, and a limited set of symbols (#, @, _, $, ¥, or £). It cannot contain embedded spaces.

Check all applications that use **select into**. Examples of select list items that require headings are:

- An aggregate function:

  ```
  avg(advance)
  ```

- Arithmetic expression:

```
colname * 2
```

- String concatenation:

```
au_lname + ", " + au_fname
```

- Built-in function:

```
substring(au_lname,1,5)
```

- Constant:

```
"Result"
```

There are three ways to specify the column heading:

- Before the expression or aggregate function, with "=":

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

- After the expression or aggregate function:

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

- After the expression or aggregate function, with **as**:

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

The column names must be valid identifiers, unless you are using the delimited identifier feature.

Changes to Correlation Name Handling

As of release 10.0, queries that include correlation names conform to the requirements of SQL92. In earlier releases, statements that specified correlation names but did not use them consistently still returned results. These statements now return errors. For example, this statement is incorrect:

```
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 this same type of correlation is included in a subquery, no error is reported, but queries may return different results. Here is an example:

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

In earlier releases, the reference to *mytable.columnC* in the subquery referred to the *mytable* in the subquery. Now, this query is a correlated subquery, and *mytable.columnC* refers to the outer table *mytable*.

If the query needs to refer to the *mytable* in the subquery, the correct statement is:

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

Subquery Changes

Several problems relating to subqueries have been fixed as of SQL Server release 10.0. This section describes old and new behaviors in detail. Examine applications that use subqueries to determine if your application relies on a behavior that has been corrected. This may be especially true of the first two items below.

The following types of subqueries have changed:

- Correlated subqueries using **in** and **any**
- Subqueries using **not in** when the subquery returns NULL values
- **in** and **any** subqueries combined with **or**
- >**all** and <**all** with subqueries that return no rows
- Subqueries that suppressed duplicate values from the outer query
- Aggregate queries with **exists** when there are duplicate values
- Correlated subqueries with **distinct** when used with **in**

In addition, support has been added for:

- The use of =**all** to introduce subqueries

- The use of aggregates in **where** clauses, if the following conditions are true:
  - The aggregate appears in a subquery that is in a **having** clause
  - The aggregated value refers only to tables named in the **from** clause of the outer query
  - The aggregated value does not refer to tables named in the **from** clause of the subquery itself

An **in** subquery (or a subquery using **any**) performs an existence check. The subquery evaluates to TRUE or FALSE. Correlated subqueries return a single TRUE or FALSE value for each row in the outer query. A subquery should not cause any row from the outer query to be returned more than once.

Subqueries using the **in** predicate used to return duplicates. The same is true for subqueries using **any**. For example, using the *pubs2* database:

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

Or:

```
select pub_name
    from publishers
    where pub_id = any
        (select pub_id
             from titles)
```

In pre-10.0 releases of Transact-SQL, these queries both returned a *pub_name* for each row in the inner query:

```
pub_name
----------------
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
```

They now return:

```
pub_name
------------------------
New Age Books
Binnet & Hardley
Algodata Infosystems
```

Subqueries Using *not in* or *all* with NULL

A subquery using the **not in** or **all** predicate returns a set of values for each row in the outer query. If the value from the outer query is not in this set, the **not in** returns TRUE. If the set returned by the subquery contains no matching value, but contains a NULL, the **not in** returns UNKNOWN. This is because NULL stands for "value unknown," and it is impossible to tell whether the value you are looking for is in a set containing an unknown value. In earlier releases, SQL Server erroneously considered this case to be TRUE.

For example, using the *pubs2* database:

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

In previous releases, this query returned:

```
pub_id
------
0736
0877
1389
```

It now returns the correct result because publishers 0877 and 1389 have rows where the price is NULL:

```
pub_id
------
0736
```

in and *any* Subqueries Using *or*

When a subquery using the **any** or **in** predicate is combined with **or** in a logical expression, the query should return rows if the subquery evaluates to FALSE, when the other clause of the logical expression evaluates to TRUE. The same is true of subqueries using **any**. For example, using the *pubs2* database:

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

Or:

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

These queries both returned no result rows in pre-10.0 SQL Server.

They now return:

```
pub_name
----------------------------
Algodata Infosystems
```

>*all* and <*all* When the Subquery Matches No Rows

When a subquery using the **>all** or **<all** predicate matches no rows, the subquery should evaluate to TRUE. Instead, it evaluated to FALSE in pre-10.0 SQL Server. For example, using the *pubs2* database:

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

This query returned no rows. It now returns all of the rows, the correct result.

Correlated Subqueries Suppressing Duplicates from Outer Query

A correlated subquery returning a data value to the outer query sometimes caused the outer query to fail to return duplicate rows if all of the columns in the outer query are also used in the subquery. For example, using the *pubs2* database:

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

In previous versions, this query returned these results:

```
pub_id
------ -----------
0736            5
0877            7
1389            6
```

It now returns:

```
pub_id
------ -----------
1389            6
1389            6
0736            5
1389            6
0877            7
0877            7
0877            7
1389            6
```

```
1389            6
1389            6
0877            7
0736            5
0736            5
0736            5
0736            5
0877            7
0877            7
0877            7
```

Aggregate Queries with *exists* Subqueries

A query that selects an aggregate and has a correlated subquery that uses the **exists** predicate produced the wrong answer if the table in the subquery's **from** list had duplicates. For example, using the *pubs2* database:

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

In pre-10.0 SQL Server, this query returned the value 18. It now returns the correct response, 3. It is similar to the first problem in this discussion, as it has to do with duplicates in an "existence-type" subquery to cause duplicates in the outer query.

Correlated Subqueries Using *distinct* and the *in* Predicate

A correlated subquery introduced with the **in** predicate and having a **select distinct** in the subquery previously returned no rows. For example, using the *pubs2* database:

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

This query returned no rows in pre-10.0 SQL Server. It now returns:

```
pub_name
----------------------------------------
New Age Books
Binnet & Hardley
Algodata Infosystems

(3 rows affected)
```

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T