What's New in SYBASE SQL Server™ Release 10.0?

SYBASE SQL Server Release 10.0

Document ID: 36440-01-1000-04

Change Level: 2

Last Revised: June 17, 1994

Principal authorship: Server Publications Group

Document ID: 36440-01-1000

This publication pertains to SYBASE SQL Server Release 10.0 of the SYBASE database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of the agreement.

Document-Back Guarantee

Sybase welcomes corrections and comments on its documents. If you mark typographical errors, formatting errors, errors of fact, or areas that need clarification in any Sybase user's manual and send copies of marked-up pages to us, we will send you a clean copy of the manual, absolutely free**.**

Send pages to the Publications Operations Department at the address below. Please include your Site ID number.

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

(510) 922-3500
Fax (510) 922-5340

Document Orders

Customers may purchase additional copies of any document or the right to make photocopies of documentation for their in-house use.

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

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

Upgrades are provided only at regularly scheduled software release dates.

Table of Contents

1. New Features in SQL Server Release 10.0

Introduction . . . 1-1
New Installation and Upgrade Utility . . . 1-2
Addition to System Databases . . . 1-2
  Benefits of Moving System Procedures . . . 1-3
  The Upgrade Process . . . 1-3
Backup/Space Management . . . 1-4
  Backup Server . . . 1-4
    Installing a Backup Server . . . 1-4
    Additional Sources of Information . . . 1-5
  Threshold Manager . . . 1-5
    Variables Allowed in Dump and Load Commands . . . 1-6
New Security Features . . . 1-6
  System Administration Roles . . . 1-7
  User Identification . . . 1-8
  Auditing . . . 1-8
Cursors . . . 1-9
Data Definition Enhancements . . . 1-9
  Integrity Constraints . . . 1-9
  Changes to Views . . . 1-10
  Schemas . . . 1-10
  Datatypes . . . 1-10
  Automatic Sequential Values Using the IDENTITY Property . . . 1-11
    Automatic Creation of IDENTITY Columns . . . 1-12
Transactions, Triggers, and Stored Procedures Changes . . . 1-12
  Data Definition Language in Transactions . . . 1-12
  Transaction State Information: @@*transtate* . . . 1-13
  New *rollback trigger* Command . . . 1-13
  Trigger Self-Recursion . . . 1-13
  Stored Procedure Size Limits Removed . . . 1-13
Query/Data Modification Changes . . . 1-14
  Datatype Conversion Changes . . . 1-14
  New Hex Conversion Functions . . . 1-14
  Subquery Changes . . . 1-14
  Grouping on *bit* Columns Allowed . . . 1-15
  Random Number Generator Improvements . . . 1-15

between Predicate Changes 1-15
select into Column Headings 1-15
Column Alias. 1-15
Full Dynamic SQL / Precompiler Support 1-15
Right Truncation of Character Strings 1-15
ANSI Compatibility 1-16
set Commands Required for ANSI Compliance. 1-16
FIPS Flagger. 1-17
SQLSTATE Messages and Codes 1-17
The *escape* Clause in the *like* Predicate 1-17
ANSI-Style Comments. 1-17
Changes to *set* Options *arithabort* and *arithignore* 1-18
Synonymous Keywords. 1-18
Chained Transactions and Isolation Levels. 1-19
Delimited Identifiers 1-19
Treatment of Nulls 1-20
Right Truncation of Character Strings 1-20
Permissions Required for *update* and *delete* Statements 1-20
Enhancements to Permissions 1-20
grant with grant option 1-21
Granting to Roles 1-21
Configurable Packet Size 1-21
Chargeback Accounting 1-21
New *dbcc* Options 1-22
kill Command Enhancements 1-22
shutdown Command Enhancements 1-22
tempdb Changes. 1-22
Additional Information on Space Usage 1-23
Error Handling. 1-23
***create index* Performance Enhancements** 1-23
Improvements to the Query Optimizer 1-24
Bulk Copy Performance Enhancements 1-24
Spanish Collating Orders 1-24
Document Reorganization. 1-25
Summary by Command Name, Type, Tables Affected 1-25
set Options 1-25
New and Changed Built-In Functions. 1-27
New System Procedures 1-28
Changes to System Procedures. 1-30

New Configuration Variables 1-30
New Database Options 1-31
New System Tables 1-32
Changes to Existing System Tables 1-33

2. Changes That May Affect Existing Applications

Introduction 2-1
New Transact-SQL Keywords 2-2
Password Changes 2-3
Addition of *sybsystemprocs* Database 2-3
Changing Permissions on System Procedures 2-4
Moving Your Own Procedures 2-4
Changing References to *master*’s System Tables 2-4
Checking for the Existence of Transactions 2-4
Configuring Open Databases 2-5
Changing Backup and *dbcc* Procedures 2-5
Changes in Master Recovery 2-5
Remote Access and the Backup Server 2-5
Changes to Dump Scripts 2-6
Changes to Renaming Databases 2-8
Datatype and Conversion Changes 2-8
Display Format Changes 2-8
Default Type Changes 2-9
Datatype Hierarchy Changes 2-9
Overflow Changes 2-10
Floating Point-to-Character Conversions 2-11
Numeric-to-Numeric Conversions Requiring Truncation 2-11
Integer-to-Character Conversions 2-11
Changes to Conversion Error Messages 2-11
Change to *between* 2-12
ANSI-Style Comments 2-12
ANSI-Compliant Permissions Requirements for *update* and *delete* 2-13
ANSI-Compliant Treatment of Nulls 2-14
Switching to Chained Transaction Mode 2-14
Using Roles in SQL Server 2-14
Repeated Table Names in *from* Clauses 2-15
Column Names Required for *select into* with Expressions 2-15
Changes to Correlation Name Handling 2-16

Subquery Changes 2-17
Subqueries Using *not in* with NULL 2-20
in and *any* Subqueries Using *or*. 2-20
> *all* and < *all* When the Subquery Matches No Rows. 2-21
Correlated Subqueries Suppressing Duplicates from Outer Query . . . 2-21
Aggregate Queries with *exists* Subqueries 2-22
Correlated Subqueries Using *distinct* and the *in* Predicate. 2-23

3. Changes in SQL Server Releases 4.8, 4.9, and 4.9.1

Introduction 3-1
Summary of Changes, SQL Server Release 4.8 3-1
Fast *create database* for Database Recovery 3-1
union Operator 3-1
New “Not Equals” Operator. 3-1
New Datatypes 3-1
Quoted Strings in Column Headings 3-2
International Features 3-2
System Tables for Alternate Language Support 3-2
New Utility Program 3-2
Changed Utility Programs 3-2
New Transact-SQL *set* Command Options. 3-2
Configuration Variables 3-3
New Global Variables 3-3
New System Procedures 3-3
Changed System Procedures 3-3
Alternate Sort Order Support. 3-3
Symmetric Multiprocessor Features 3-3
Changes to System Tables 3-3
New Built-In Functions 3-4
Changes to Error Log Format 3-4
New Configuration Variables 3-4
Summary of Changes, SQL Server Release 4.9 3-4
Multibyte Character Set Features. 3-4
Character Set Conversion 3-5
Changing Sort Orders and Character Sets 3-5
New Datatypes. 3-5
New System Table 3-5
Enhancements to *print* and *raiserror* 3-5
Changed String Function 3-6

New Functions 3-6
New Global Variables 3-6
Configuration Variables 3-6
New System Procedures 3-6
dbcc Command Enhancements 3-7
Summary of Changes, SQL Server Release 4.9.1 3-7
dbcc Command Enhancements 3-7
New System Procedures 3-7
Configuration Variable. 3-8

Index

List of Tables

Table 1-1: *set* Options for ANSI Compliance 1-16
Table 1-2: ANSI-Compatible Keyword Synonyms 1-18
Table 1-3: New *set* Options 1-25
Table 1-4: New and Changed Built-In Functions 1-27
Table 1-5: New System Procedures 1-28
Table 1-6: Chargeback Accounting System Procedures 1-29
Table 1-7: Changes to System Procedures 1-30
Table 1-8: New Configuration Options 1-30
Table 1-9: New *sp_dboption* Database Options 1-31
Table 1-10: Changes to Existing System Tables 1-33
Table 2-1: New Reserved Words 2-2
Table 2-2: Changes in Output Using Numeric Literals 2-9
Table 2-3: Changes to Datatype Hierarchy 2-10
Table 2-4: New Type Conversion Error Messages 2-12
Table 2-5: ANSI Permissions for *update* and *delete* 2-13

1 New Features in SQL Server Release 10.0

Introduction

This chapter describes the new features for SYBASE SQL Server™ Release 10.0. These new features include:

- New Installation, Upgrade and Configuration Utility
- Addition to System Databases
- Backup / Space Management
- New Security Features
- Cursors
- Data Definition Enhancements
- Trigger, Transaction, and Stored Procedure Changes
- Query / Data Modification Changes
- ANSI Compliance Changes
- System Administration Changes
- Error Handling
- **create index** Performance Enhancements
- Query Optimizer Enhancements
- Bulk Copy Performance Enhancements
- Spanish Language Sort Orders
- Document Reorganization

The last section of this chapter provides a summary of changes to the following commands and system tables:

- **set** Options
- New and Changed Built-In Functions
- New and Changed System Procedures
- New Configuration Options
- New Database Options
- New System Tables
- Changes to System Tables

New Installation and Upgrade Utility

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 a 4.8 or later release to Release 10.0
- Modify an existing 10.0 SQL Server to install and activate auditing
- Install languages and character sets
- Configure your default language, sort order or character set
- Install other Sybase 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 *SYBASE SQL Server Installation 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. The space requirements for the new system tables and system procedures would nearly double the size requirement for the *master* database. Since *master* must reside entirely on a single device, many customers would face serious problems (in some cases requiring disk repartitioning and moving *master*).

Release 10.0 solves this problem by removing the largest space users from *master* in a way that is transparent to ordinary users of SQL Server and requires a minimum of work for the System Administrator who upgrades SQL Server.

In Release 10.0, the SYBASE system procedures are stored in a new database called *sybsystemprocs*, rather than in the *master* database. When your installation or upgrade is complete, Sybase system procedures reside in the new 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 shrinks the space used in *master* dramatically. The size of the *master* database will not have to change for a long time.
- It creates greater flexibility for users. Users are able to 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 will be updated rarely and can be easily restored.
- It keeps backups of *master* smaller. 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*. **sybinit** 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 processes increases the number of open databases allowed on your SQL Server.

See Chapter 2 of this manual for information on moving your local procedures from *master* to *sybsystemprocs*, and other system administration information relating to this change.

Backup/Space Management

Backup Server

The Open Server-based Backup Server utility now handles all dumps and loads for SQL Server. Dumps and loads now work more quickly, 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.
- Multi-file / multi-volume 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 labelling and handling.
- Platform-specific tape handling options support dump / load command syntax specification for volume naming, dismount and load control, tape density, block size, tape capacity, days to retain, initialization, file naming for multi-dump 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 syntax. The only exception is that **dump database** and **dump transaction** commands cannot be included in a user-defined transaction in Release 10.0.

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 earlier releases using 10.0 dump and load commands: pre-10.0 dumps are not compatible with 10.0 SQL Server.

The *SYBASE SQL Server Installation Guide* contains information about installing Backup Servers, and the *System Administration Guide Supplement* for your platform contains information on supported dump devices and information about platform-specific issues.

Additional Sources of Information

The new syntax for the **dump database, load database**, **dump transaction**, and **load transaction** commands is documented in Volume 1 of the *SQL Server Reference Manual*. **sp_volchanged** is documented in Volume 2 of the *SQL Server Reference Manual*.

Information about starting and stopping Backup Servers during routine operations can be found in **startserver** in the *SQL Server Utility Programs* for your operating system, **shutdown** in Volume 1 of the *SQL Server Reference Manual*, and your *System Administration Guide Supplement*. See **backupserver** in the *SQL Server Utility Programs* for information about flags for the **backupserver** executable.

See Chapter 7, ‘‘Developing a Backup and Recovery Plan’’, in 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 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. See Volume 2 of the *SQL Server*

Reference Manual for specific syntax information. Chapter 10, "Managing Free Space with Thresholds", in the *System Administration Guide* provides additional examples.

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

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 under pre-10.0 SQL Servers.

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 **sp_addthreshold**, **sp_dboption**, and Chapter 10, "Managing Free Space with Thresholds", of the *System Administration Guide* for examples.

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 Volume 1 of the *SQL Server 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, and a login account may possess more than one role. Roles can be granted permissions on objects and commands, similar to groups. Details on roles can be found in Chapter 2, ‘‘Roles in SQL Server’’, in the *System Administration Guide*.

When Release 10.0 of SQL Server is installed, it still has the default “sa” account, which has 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 ‘‘Using Roles in SQL Server’’ on page 2-14 for more information.

User Identification

There are now a variety of mechanisms SQL Server uses to positively identify an individual user and to enforce accountability and security, such as:

- 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.

New sections in Volume 1 of the *SQL Server Reference Manual* called "Roles" and "Login Management" provide an overview of the system procedures and other commands used to manage these features.

Auditing

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 Chapter 16, ''Auditing'', in the *System Administration Guide*.

Cursors

SQL Server Release 10.0 provides full support for cursors. The following cursor commands are documented in Volume 1 of the *SQL Server Reference Manual*:

declare cursor
fetch
open
close
deallocate

A new topic called "Cursors" in Volume 1 of the *SQL Server Reference Manual* provides an overview of cursors. 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.

Chapter 15, "Cursors: Accessing Data Row by Row", in the *Transact-SQL User's Guide* provides examples of cursor use, including cursors in stored procedures.

Data Definition Enhancements

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 insure unique values in a column
- Referential integrity constraints to guarantee that a primary key exists in another table if 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 **create table** and **alter table** in Volume 1 of the *SQL Server Reference Manual* for the new syntax. Chapter 6, “Creating Databases and Tables”, in the *Transact-SQL User’s Guide* describes how to use the constraints along with 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 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 Volume 1 of the *SQL Server Reference Manual*, or Chapter 8, “Views: Limiting Access to Data”, in the *Transact-SQL User’s Guide* for more information and examples.

Schemas

The new **create schema** command allows you to create several objects and grant permissions on those objects in a single statement, which can be committed or rolled back as a unit. See **create schema** in Volume 1 of the *SQL Server Reference Manual*.

Datatypes

Release 10.0 of SQL Server provides these new datatypes:

- *numeric* or *decimal* (*dec*) specify exact numeric values, with specified precision and scale to indicate the 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 *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 Chapter 2, ‘‘Changes That May Affect Existing Applications’’, for information on how this can affect existing applications.

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

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

See “Datatypes” in Volume 1 of the *SQL Server Reference Manual* or Chapter 6 in the *Transact-SQL User’s Guide* for more information.

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 IDENTITY Columns and **create table**, **alter table**, **insert**, **select**, **create view** in Volume 1 of the *SQL Server Reference Manual*.

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 "Identity Columns" in Chapter 3 of the *SQL Server Reference Manual*, Volume 1.

Transactions, Triggers, and Stored Procedures Changes

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**, **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 "Transactions" in Volume 1 of the *SQL Server Reference Manual*.

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*.

See "Transactions" in Volume 1 of the *SQL Server Reference Manual*. "Variables" in Volume 1 of the *SQL Server Reference Manual* also provides a list of the status values.

New *rollback trigger* Command

The rollback feature has been expanded to include **rollback trigger.** Now, you can choose to roll back an entire transaction using **rollback transaction**, or to roll back statements up to the first data modification that fired a trigger using **rollback trigger**. See **rollback trigger** and "Transactions" in Volume 1 of the *SQL Server Reference Manual* for more information.

Trigger Self-Recursion

A new **set** option, **self_recursion**, allows triggers to fire self-recursively: 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 Volume 1 of the *SQL Server Reference Manual* for more information.

Stored Procedure Size Limits Removed

Release 10.0 of SQL Server 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 16 Mb. This increase also applies to objects whose text is stored in the *syscomments* system table, such as triggers, views, defaults, rules, and constraints.

Query/Data Modification Changes

Datatype Conversion Changes

Changes were made to datatype conversion rules for this release 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 Chapter 2, "Changes That May Affect Existing Applications", for information on how these changes may affect existing applications. The new hierarchy is included in the "Datatypes" section of Volume 1 in the *SQL Server Reference Manual*.

Additional information about datatype conversions is provided in "Datatype Conversion Functions" in Volume 1 of the *SQL Server Reference Manual* and Chapter 9, "Using the Built-In Functions in Queries" in the *Transact-SQL User's Guide*.

New Hex Conversion Functions

Two new functions provide platform-independent conversions between integer values and hexadecimal strings. See "Datatype Conversion Functions" in Volume 1 of the *SQL Server Reference Manual* and Chapter 9, "Using the Built-In Functions in Queries" in the *Transact-SQL User's Guide* for information about **inttohex** and **hextoint**.

Subquery Changes

The behavior of certain subqueries has changed. See Chapter 2, "Changes That May Affect Existing Applications" 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 “Mathematical Functions” in Volume 1 of the *SQL Server Reference Manual* for more information.

between Predicate Changes

In some cases, the **between** predicate in Transact-SQL returned results whether the values supplied were in the proper order (lower number first after the predicate) or 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 Chapter 2 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. Full documentation can be found in *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 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 ANSI behavior. See **set** in Volume 1, Chapter 1 of the *SQL Server Reference Manual* for more information.

ANSI Compatibility

SQL Server Release 10.0 meets SQL89 / FIPS 127-1 and SQL92 / FIPS 127-2 standards. 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. Since some of ANSI behavior is not compatible with existing SQL Server applications, Transact-SQL provides **set** options that allow you to toggle these behaviors.

set Commands Required for ANSI Compliance

Compliant behavior is enabled by default for all Embedded SQL™ precompiler applications. Other applications needing to match ANSI behavior can use option settings in *Table 1-1: set Options for ANSI Compliance* for entry level SQL92 compliance.

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**

Table 1-1: **set** *Options for ANSI Compliance*

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 Chapter 1 of the *SQL Server Reference Manual,* Volume 1.

FIPS Flagger

For customers writing applications that must conform to the ANSI standard, 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 Volume 1, Chapter 1 of the *SQL Server 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 1-1 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 ANSI **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 "Wildcard Characters" in Volume 1, Chapter 3 of the *SQL Server Reference Manual*.

ANSI-Style Comments

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

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

This syntax conflicts with the subtraction of a negative number. See Chapter 2, "Changes That May Affect Existing Applications", 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* type. 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 ensure that they are still producing the desired behavior.

Synonymous Keywords

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

Pre-Release 10.0 Transact-SQL	Additional Syntax
tran **transaction**	**work**
any	**some**
grant all	**grant all privileges**
revoke all	**revoke all privileges**

Table 1-2: ANSI-Compatible Keyword Synonyms

Pre-Release 10.0 Transact-SQL	**Additional Syntax**
max (*expression*)	**max** ([**all** | **distinct**]) *expression*
min (*expression*)	**min** ([**all** | **distinct**]) *expression*
user_name built-in function	**user** keyword

Table 1-2: ANSI-Compatible Keyword Synonyms (continued)

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 page 1-11 of this document for a list of synonymous datatypes.

Chained Transactions and Isolation Levels

SQL Server now provides ANSI-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 Volume 1 of the *SQL Server 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 "Transactions" in Volume 1 of the *SQL Server Reference Manual* for more information.

Delimited Identifiers

SQL Server now supports delimited identifiers for table, view, and column names. Delimited identifiers are object names enclosed within 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 ANSI-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 ANSI compatibility. Set this option **on** to prohibit silent truncation and enforce ANSI 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, though not to groups, roles, or **public**. A corresponding clause, **cascade**, for the **revoke** command, allows the privilege holder to revoke permissions from all users who obtained it via **with grant option**. See **grant** and **revoke** in Volume 1, Chapter 1 of the *SQL Server Reference Manual* for syntax information. Chapter 5, ‘‘Managing User Permissions’’, of the *System Administration Guide* provides 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 transferring large amounts of data in or out of SQL Server. See Chapter 12, ‘‘Fine-Tuning Performance and Operations’’ of 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 your operating system for information about using these options. Open Client™ Client-Library documentation includes 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 VMS.

See **sp_clearstats** and **sp_reportstats** in Volume 2 of the *SQL Server Reference Manual* for information on getting reports and restarting the accounting interval. See **sp_configure** for information on the **cpu flush** and **i/o flush** configuration variables 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 certain 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 only actually terminate when the processes stopped sleeping. Now, these sessions can be terminated immediately. The **sp_who** system procedure now displays different status values for these sessions. For more information, see **kill** in Volume 1, Chapter 1 of the *SQL Server Reference Manual* or Chapter 11, ''Diagnosing System Problems'' of the *System Administration Guide.*

shutdown Command Enhancements

The **shutdown** command, used to shutdown 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 **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 “Temporary Tables” in Volume 1, Chapter 3 of the *SQL Server Reference Manual* for more information.

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 name of the tables and columns causing an error, along with the error number and text. See **raiserror** in Volume 1 of the *SQL Server Reference Manual*.

create index Performance Enhancements

Internal improvements have enhanced the speed of the **create index** command. In addition, 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 an extent at a time, thereby making more efficient use of the disk and increasing the speed of the **create index** command. If you wish 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 page-at-a-time disk I/O. For best performance, you should 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 12, “Fine-Tuning Performance and Operations” for information about the new **sp_configure** option, **extent i/o buffers.** The section contains suggestions for choosing the correct buffer size.

Improvements to the Query Optimizer

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 would often underestimate 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, the first and second columns, 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 under earlier versions 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

Internal improvements, transparent to users, result in faster bulk copy speeds. In addition, configurable packet sizes can increase bulk copy throughput. See **bcp** in the *SQL Server Utility Programs* manual and Chapter 12, "Fine-Tuning Performance and Operations", 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 17, "Language, Sort Order, and Character Set Issues" in the *System Administration Guide*. Instructions for installing a new SQL Server using a Spanish collating sequence are in your *SYBASE SQL Server Installation Guide*.

Document Reorganization

The SQL Server documentation set has been reorganized for this release.

- The *Commands Reference Manual* has been split into two volumes:
  - Transact-SQL commands, functions, and special topics (such as Locking and Transactions) are now documented in Volume 1 of the *SYBASE SQL Server Reference Manual.*
  - The SYBASE system procedures and catalog stored procedures are now documented in Volume 2 of the *SYBASE SQL Server Reference Manual*.
- The *SYBASE SQL Server Utility Programs* now documents the utility programs for your operating system.
- The *Transact-SQL User's Guide* has been reorganized, with the first part tailored to new SQL users and the more advanced and complex concepts presented in the second part.

Summary by Command Name, Type, Tables Affected

set Options

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

Option	Use
set ansinull {on | off}	Toggles ANSI treatment of NULL-valued operands in equality (=) and inequality (!=) comparisons. When **on**, treatment is ANSI '89 compliant.
set ansi_permissions {on | off}	Determines whether ANSI-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.

Table 1-3: New set *Options*

Option	Use
set chained {on | off}	Toggles the chained transaction mode in SQL Server.
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. ANSI 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 ANSI 89 SQL.
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 if silent truncation of character strings is allowed. When **on**, treatment is ANSI '89 compliant
set transaction isolation level {1|3}	Sets the transaction isolation level. An isolation level of 3 is equivalent of doing **select with holdlock.**

Table 1-3: New set *Options (continued)*

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:

Name	Function
col_name	Changed to accept a database name as an optional third parameter.
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 ANSI compatible system function returns the user name.

Table 1-4: New and Changed Built-In Functions

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

See Volume 1, Chapter 2 of the *SYBASE SQL Server 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:

Name	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	Establishes auditing of different types of events within a database, or of references to objects within that database from another database.
sp_auditlogin	Audits a user's attempts to access tables and views, and/or the text of a user's commands.
sp_auditobject	Establishes auditing of accesses to tables and views.
sp_auditoption	Enables and disables system-wide auditing and global audit options.
sp_auditsproc	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 which are active.
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 on all thresholds in the current database or all thresholds for a particular segment.

Table 1-5: New System Procedures

Name	Function
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**.

Table 1-5: New System Procedures (continued)

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

Name	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.

Table 1-6: Chargeback Accounting System Procedures

See Volume 2 of the *SQL Server Reference Manual* for information on these procedures.

Changes to System Procedures

The following system procedures have been changed in this release:

Name	Changes
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.

Table 1-7: Changes to System Procedures

New Configuration Variables

There are new configuration variables for use with **sp_configure**:

Option	Function
additional netmem	Memory added for use with variable packet sizes
audit queue size	Number of audit records in the audit queue
cpu flush	Used to configure chargeback accounting (new on non-OpenVMS platforms).
default network packet size	Default packet size.
extent i/o buffers	Allocates additional memory to be used for buffering data pages during **create index** and **order by** read / writes to disk.

Table 1-8: New Configuration Options

Option	Function
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 flush	Used to configure chargeback accounting (new on non-OpenVMS platforms).
maximum network packet size	Maximum allowable packet size.
password expiration interval	Sets password expiration time.

Table 1-8: New Configuration Options (continued)

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

New Database Options

There are new 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 ANSI 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; you must include the column name in the select list to retrieve it.

Table 1-9: New sp_dboption *Database Options*

Option	Function
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.

Table 1-9: New sp_dboption *Database Options (continued)*

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

There are four new system tables in all databases:

- *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).

These are fully documented in Appendix B, “The System Tables”, included in both the *System Administration Guide* and the *SQL Server Reference Manual.*

Changes to Existing System Tables

The following system tables have changed in this release:

Table	Changes
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 VMS 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*.

Table 1-10: Changes to Existing System Tables

2 Changes That May Affect Existing Applications

Introduction

This chapter describes the changes to Release 10.0 of SQL Server that may affect existing applications. These changes are:

- New Transact-SQL keywords
- Password changes
- Addition of *sybsystemprocs* database
- Remote access configured “on” in all existing databases for Backup Server
- Changes to dump scripts
- Changes to renaming databases
- Datatype changes:
  - Display Format Changes
  - Changes to the datatype hierarchy and defaults
  - Overflow changes, including float-to-character, numeric-to-numeric, and integer-to-character conversions.
- Change to **between**
- Addition of ANSI-style comments using “--”
- Addition of ANSI-compliant permissions requirements for **update** and **delete** statements
- Addition of ANSI-compliant treatment of nulls when **set ansinull** is **on**
- Chained transaction mode and its effect on stored procedures
- Role changes that may affect scripts
- Repeated table names in the **from** clause of an **update** or **select** statement now causes an error
- Column names required for **select into** when the select list contains expressions or aggregates
- Changes to correlation name handling
- Changes to subquery behavior

In addition, you may want to review applications using “browse mode”, to take advantage of cursors.

New Transact-SQL Keywords

The following keywords are new "reserved" words in 10.0 SQL Server. They cannot be used as object names or column names.

arith_overflow	identity_insert	references
at	isolation	replace
authorization	key	role
cascade	level	rows
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	

Table 2-1: New Reserved Words

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 to Release 10.0, 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. For information on running this procedure before you upgrade, see your *SYBASE SQL Server Installation Guide*.

If you choose to leave some of these reserved words as identifiers until after you upgrade to 10.0, you can run the procedure **sp_checkreswords** at any time. See **sp_checkreswords** in Volume 2 of the *SQL Server 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 6-character new password.

You can globally force users to change passwords within a specified time period using the **password expiration level** 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 Chapter 12, "Fine-Tuning Performance and Operations", in the *System Administration Guide* for more information.

Addition of *sybsystemprocs* Database

In Release 10.0, all Sybase-provided system procedures are now 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 variable, 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 change.
- Recovery of *master* database and system databases has 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 of the tables 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 that all of 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 Chapter 9, "Backing Up and Restoring the System Databases", in the *System Administration Guide* for more information.

Remote Access and the Backup Server

All dump and load operations in Release 10.0 of SQL Server are executed through remote procedure calls to the Backup Server. To allow SQL Server to communicate with the Backup Server, the configuration variable **remote access** must be turned on. Leave this variable turned on unless you have security concerns. You must be a System Security Officer to set **remote access**.

Turning on **remote access** 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 wish to leave **remote access** disabled except during a dump or load, you must always issue the following command before issuing the dump or load command:

```
sp_configure "remote access", 1
```

After you complete the dump, turn **remote access** off again with the command:

```
sp_configure "remote access", 0
```

The **remote access** configuration variable 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 changed in Release 10.0 of SQL Server. 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 VMS) are now prohibited.

Here are some guidelines for backing up your databases immediately after upgrading to 10.0:

- 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, quarter-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 will overwrite the existing tape; a negative response initiates a request for a volume change, and tests are repeated on the new volume.
- If you use multi-file tape media, and do not change your dump scripts, the dump will be appended to the existing files on the tape.

- If you want to overwrite existing tapes that have ANSI labels, you must append the **with init** clause to existing dump commands:

  ```
  dump 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:

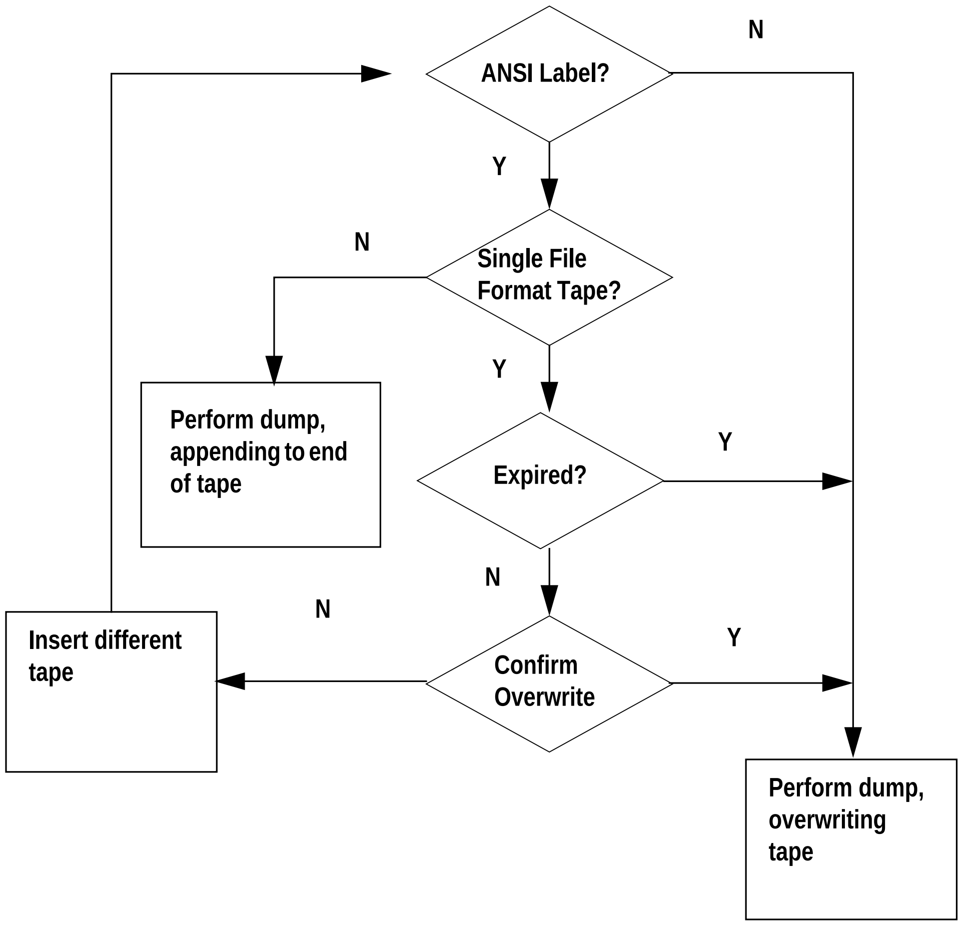

Figure 2-1: Checking Tape Format and Expiration Dates for Dump Commands

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 Volume 2 of the *SQL Server Reference Manual* for more information about renaming databases.

Datatype and Conversion Changes

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 six 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:

SQL Statement	Pre-10.0 Servers	10.0
`select 2147483648`	integer overflow	2147483648.
`select 2*1.12345678`	2.246914	2.24691356.
`select 2.8 * 5`	14.000000	14.0

Table 2-2: Changes in Output Using Numeric Literals

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 in Release 10.0. Specifically, *float* and *numeric* computations combined with *money* or *smallmoney*.

Datatypes	Pre-10.0 Servers	10.0
money and *numeric*	N/A	*numeric*
money and *float*	*money*	*float*

Table 2-3: Changes to Datatype Hierarchy

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 off**, SQL Server truncates the query results and continues processing. This option is set **on** by default.

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:

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.

Table 2-4: New Type Conversion Error Messages

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*. In 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.

ANSI-Style Comments

To comply with ANSI 89, SQL Server now supports ANSI-style comments, which consist of two dash or 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 "-2" in parentheses:

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

ANSI-Compliant Permissions Requirements for *update* and *delete*

SQL Server now supports ANSI-compliant permissions requirements for **update** and **delete** statements. The following table summarizes these requirements:

	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

Table 2-5: ANSI Permissions for update *and* delete

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.

ANSI-Compliant Treatment of Nulls

SQL Server now supports ANSI ‘89 -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 ANSI chained transaction mode. In Release 10.0, all transactions will be tagged to identify which mode they use. If you intend to convert existing SQL Server applications to use chained transaction mode, see “Transactions” in Volume 1 of the *SQL Server Reference Manual* and **sp_procxmode** in Volume 2 for information about changing transaction modes.

Using Roles in SQL Server

See ‘‘System Administration Roles’’ on page 1-7 for a description of the new roles. There are two ways to use roles in the SQL Server:

- As before, use the “sa” account for system administration tasks. When SQL Server is installed, this account has 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 ANSI 89, the following construction is not allowed:

```
select * from table1, table1
    [where_clause]
```

In earlier releases, SQL Server handled the **select** by returning only the first table and ignoring the self-join. In 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:

```
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 ANSI 89 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 heading 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. In 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.

You should 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 an "=":

```
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

In Release 10.0, queries that include correlation names conform to ANSI requirements. 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 in Release 10.0 of SQL Server. 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. This may especially be 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 rows from the outer query to be returned more than once each.

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
```

If you have applications which depend on the pre-10.0 Transact-SQL behavior, use the following **set** command until you can rewrite the existing queries as joins, or add indexes to the tables the queries operate on:

```
set dup_in_subquery on
```

◆ ***WARNING!***

***set dup_in_subquery on* is provided only as an upgrade path to SQL Server Release 10.0. It will not be supported in future releases. See the *Release Bulletin* for more information.**

Subqueries Using *not in* with NULL

A subquery using the **not in** 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 it does contain 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're 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:

```
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)
```

3 Changes in SQL Server Releases 4.8, 4.9, and 4.9.1

Introduction

This chapter lists the changes to Releases 4.8, 4.9, and 4.9.1 of SQL Server. This information is provided mainly as a convenience for customers who have not upgraded to one of these earlier versions.

Summary of Changes, SQL Server Release 4.8

This summary describes the significant changes in SQL Server Release 4.8.

Fast *create database* for Database Recovery

The new **for load** option for the **create database** command speeds up database creation for recovery from media failure or for moving databases. See **create database** in Volume 1 of the *SQL Server Reference Manual*, or the *System Administration Guide.*

union Operator

Transact-SQL now provides the **union** operator. It allows you to combine the results of two or more queries into a single result set. See **union** in Volume 1 of the *SQL Server Reference Manual.*

New "Not Equals" Operator

The new negative comparison operator <> (not equal to) is the same as the != operator. See "Expressions" in Volume 1 of the *SQL Server Reference Manual.*

New Datatypes

smallmoney, smalldatetime, and *real* are smaller versions of *money, datetime,* and *float*, respectively. They require 4 bytes of storage, rather than the 8 bytes required by their larger counterparts. See "Datatypes" in Volume 1 of the *SQL Server Reference Manual.*

Quoted Strings in Column Headings

Except in **create table**, **create view**, and **select into** statements, column headings now can include any characters, including blanks and SQL Server keywords, if the column heading is enclosed in quotes.

International Features

SQL Server Release 4.8 contains all of the enhancements featured in SQL Server Release 4.2, the International Release. See the *System Administration Guide* and the *System Administration Guide Supplement* for your platform, the indexed entries in *SQL Server Reference Manual*, and *Transact-SQL User's Guide*, for more information on these features.

System Tables for Alternate Language Support

SQL Server now supports multiple alternate languages simultaneously, through new tables, *syscharsets* and *syslanguages*, and new columns, *langid* (*sysmessages*), *language* (*syslogins*), *csid*, and *soid* (*sysindexes*).

New Utility Program

The **langinstall** utility program installs alternate languages on a server. See the *SQL Server Utility Programs* manual for your operating system.

Changed Utility Programs

bcp, **defncopy**, and **isql** now support alternate languages and non-ISO 8859-1 terminals. The **console** program, discontinued in Release 10.0, also supported alternative languages in earlier releases. See the *SQL Server Utility Programs* manual for your operating system.

New Transact-SQL set Command Options

datefirst, **dateformat**, and **language** options to the **set** command change the first weekday, date input format, and alternate language in the current session. See **set** in Volume 1 of the *SQL Server Reference Manual*.

Configuration Variables

See **sp_configure** in Volume 2 of the *SQL Server Reference Manual* for information about these new configuration options.

New Global Variables

@@language is the name of the language currently in use. *@@langid* is the number of the language currently in use.

New System Procedures

sp_addlanguage, **sp_defaultlanguage**, **sp_droplanguage**, **sp_helplanguage**, and **sp_setlangalias** manage alternate languages.

sp_checknames reports names of objects that contain characters outside the 7-bit ASCII character set.

sp_helpsort reports the server's default sort order.

sp_recompile recompiles a table's stored procedures and triggers.

Changed System Procedures

sp_addlogin now assigns a default language when adding a user.

Alternate Sort Order Support

SQL Server can now be installed with a sort order other than the standard binary sort.

Symmetric Multiprocessor Features

Release 4.8 of the SQL Server is explicitly designed to take full advantage of the capabilities of symmetric multiprocessor (SMP) systems. See Chapter 14, "Managing Multiprocessor Servers", in the *System Administration Guide* and the *System Administration Guide Supplement* for your platform for more information.

Changes to System Tables

A new system table, *sysengines*, allows the System Administrator to monitor engines in the SMP environment.

Four system tables, *sysconfigures*, *sysindexes*, *syslocks*, and *sysprocesses*, have been modified for SMP.

New Built-In Functions

Four new built-in Transact-SQL functions give information about page allocations to a table or index. **sp_spaceused** uses these functions to provide information about database space utilization. The functions are: **data_pgs**, **reserved_pgs**, **used_pgs**, and **rowcnt**.

Changes to Error Log Format

Error log entries now show the engine involved for each log entry.

The date format is now yy/mm/dd.

The time is now displayed in 24-hour format.

Seconds and hundredths of a second have been added to the time format.

New Configuration Variables

The following configuration variables were added in 4.8:

default sortorder

default language

language in cache

max online engines

min online engines

See **sp_configure** in Volume 2 of the *SQL Server Reference Manual* or the *System Administration Guide* for more information.

Summary of Changes, SQL Server Release 4.9

This summary lists the significant product changes for Release 4.9 of SQL Server:

- A wide set of changes to support multibyte character sets
- New options to the **dbcc** command

Multibyte Character Set Features

SQL Server now supports multibyte character sets, including EUC-JIS, SHIFT-JIS, and DEC-Kanji, for use in Asian installations. The following changes and features were introduced to provide flexible language support, but some of them, such as the **print/raiserror** changes and the message-related system procedures, have much wider applications for SQL Server users.

Character Set Conversion

SQL Server supports conversion among a variety of character sets. Character set conversion is initiated using **set char_convert** within a session or by using the **-J** flag with the utilities **bcp**, **defncopy**, and **isql**.

Changing Sort Orders and Character Sets

The System Administrator can now change the default, installed sort order, or character set used by SQL Server. See the *System Administration Guide Supplement* for your platform.

New Datatypes

The national character datatypes *nchar* and *nvarchar* allow a user to define a column length as containing a specific number of multibyte characters. The number of bytes in a character is contained in the new global variable *@@ncharsize*. See "Datatypes" in Volume 1 of the *SQL Server Reference Manual.*

New System Table

The table *sysusermessages*, in all databases, stores user-defined messages for user stored procedures. Messages can be added to this table with the new **sp_addmessage** system procedure and dropped with **sp_dropmessage**. Messages can be retrieved with **sp_getmessage.** Combined with the enhancements to **print** and **raiserror** described below, application designers can now easily make their stored procedures produce error messages in a user's default language.

Enhancements to **print** *and* **raiserror**

print and **raiserror** now recognize formatted output with arguments. Format strings can contain up to 20 unique place holders in any order. To allow reordering of the arguments when format strings are translated to a language with a different grammatical structure, the place holders are numbered. See **print** or **raiserror** in Volume 1 of the *SQL Server Reference Manual.*

Changed String Function

patindex string function now returns a value in bytes **or** in characters representing the starting position of the specified character expression. See “String Functions” in Volume 1 of the *SQL Server Reference Manual*.

New Functions

valid_name determines whether a string is legal for use in an identifier.

char_length determines the number of characters in a character expression.

These functions are documented under “String Functions” in Volume 1 of the *SQL Server Reference Manual*.

New Global Variables

@@char_convert indicates whether character set conversion is in effect.

@@client_csname contains the client’s character set name. *@@client_csid* contains the client’s character set ID.

@@maxcharlen contains the maximum length of a multibyte character in SQL Server’s default character set.

@@ncharsize contains the average length, in bytes, of a national character.

Configuration Variables

The configuration variable **default character set id** specifies the number of the default character set SQL Server uses. See **sp_configure** in Volume 2 of the *SQL Server Reference Manual* or Chapter 12, ‘‘Fine-Tuning Performance and Operations’’, in the *System Administration Guide* for more information.

New System Procedures

sp_indsuspect determines whether an index needs to be rebuilt after a sort order change.

sp_addmessage, **sp_getmessage**, and **sp_dropmessage** manage user-defined messages for use in stored procedures.

dbcc Command Enhancements

dbcc (Database Consistency Checker) has two new options for use when SQL Server's sort order or character set is changed: **dbcc reindex** and **dbcc fix_text. dbcc reindex** checks and rebuilds character indexes which may have been invalidated by a sort order change. **dbcc fix_text** calculates the statistics needed to manage text values after changing to a multibyte character set.

Summary of Changes, SQL Server Release 4.9.1

This summary lists the significant product changes for Release 4.9.1 of SQL Server. Release 4.9.1 is a maintenance release.

dbcc Command Enhancements

dbcc (Database Consistency Checker) has two new options to use when checking database integrity: **dbcc tablealloc** and **dbcc indexalloc**. They perform the same checks as **dbcc checkalloc** but on individual tables and indexes. These options provide faster consistency checking, and can automatically correct certain kinds of consistency problems. For more information see Chapter 11, "Diagnosing System Problems" in the *System Administration Guide*.

New System Procedures

sp_syntax displays the syntax of Transact-SQL statements, system procedures, utilities, and DB-Library™ routines. (For Release 10.0, Client Library is also included.) See the *System Administration Guide Supplement* for information on installing the *sybsyntax* database and Volume 2 of the *SQL Server Reference Manual* for information on using the **sp_syntax** system procedure.

Twelve new system procedures allow users easy access to information about database gateways as well as SQL Server. These stored procedures are compatible with the catalog interface for the Open Database Connectivity (ODBC) API. The procedures are: **sp_column_privileges**, **sp_columns**, **sp_databases**, **sp_datatype_info**, **sp_fkeys**, **sp_pkeys**, **sp_server_info**, **sp_sproc_columns**, **sp_statistics**, **sp_stored_procedures**, **sp_table_privileges**, **sp_tables**. See Volume 2 of the *SQL Server Reference Manual* for information on using these procedures.

Configuration Variable

The new configuration variable **stack size** specifies the size of SQL Server's stack. See Volume 2 of the *SQL Server Reference Manual* or Chapter 12, ‘‘Fine-Tuning Performance and Operations’’, in the *System Administration Guide.*

Index

Symbols

- (minus signs)
  as comment marker 1-17, 2-12
@@*char_convert* global variable 3-6
@@*client_csname* global variable 3-6
@@*identity* global variable 1-11
@@*langid* global variable 3-3
@@*language* global variable 3-3
@@*maxcharlen* global variable 3-6
@@*ncharsize* global variable 3-5, 3-6
@@*transtate* global variable 1-13

A

abort tran on log full
  sp_dboption 1-31
Aggregate functions 1-20
  column headings and 1-15
all predicate 2-21
allow nulls by default
  sp_dboption 1-31
alter table 1-9
Alternate languages 3-2
ANSI compatibility
  set options for 1-16
any predicate 2-20
arithabort option
  numeric_truncation 2-11
as keyword 1-15
Auditing 1-8
  system tables 1-32
auto identity
  sp_dboption 1-31

B

Backup changes 2-6
Backup Server utility 1-4
  shutting down 1-22
Batch size limits 1-13
Batches and **create view** 1-10
bcp utility 1-21
between predicate change 1-15, 2-12
Built-in functions
  overview 1-27
Bulk copy 1-24

C

cascade option to **revoke** command 1-21
Catalog stored procedures 1-2, 3-7
Chained transactions 1-19, 2-14
char_length 3-6
Character set
  conversion 3-5, 3-6
  name 3-6
Character set default 3-6
Character string truncation 1-15, 1-20
Chargeback accounting 1-21
Check constraints 1-9
close 1-9
col_name function 1-27
Collating sequence 1-24
Column
  alias 1-15
  headings 1-15, 2-15, 3-2
Comments, ANSI-style 1-17, 2-12
Configuration variables. *See* **sp_configure**
console utility program 1-4, 3-2
Constraints 1-9
Conversion
  changes 1-14
  error message 2-11
  functions 1-27
convert changes 2-8
Correlated subqueries 2-21 to 2-23
Correlation names 2-16
create database 3-1
create index 1-23
create schema 1-10
create table 1-9

create view 1-10
  and batches 1-10
  and **distinct** 1-10
  with check option 1-10
Cursors 1-9

D

Data definition language in transactions 1-12
data_pgs function 3-4
Databases
  limit of 1-3
  master 1-2
  sybsystemprocs 1-2
Datatypes 1-14, 2-8
  bit 1-15
  char varying 1-11
  character 1-11
  conversion 1-14
  dec 1-11
  decimal 1-10, 1-11
  default length 1-10
  double precision 1-10, 1-11
  float 1-10, 1-11
  hierarchy 1-14
  hierarchy changes 2-9
  national char 1-11
  national char varying 1-11
  national character 1-11
  national character varying 1-11
  nchar 3-5
  nchar varying 1-11
  numeric 1-10
  nvarchar 3-5
  real 3-1
  smalldatetime 3-1
  smallmoney 3-1
Date input format 3-2
dbcc checkalloc
  fix option 1-22
dbcc checkdb
  skip_ncindex option 1-22
dbcc checktable
  skip_ncindex option 1-22
dbcc indexalloc 3-7
dbcc tablealloc 3-7
ddl in tran
  sp_dboption 1-12, 1-32
deallocate 1-9
decimal datatype 1-10
declare cursor 1-9
Default character set 3-6
default character set id configuration variable 3-6
Default language 3-3
default language configuration option 3-4
Default sort order 3-3
default sortorder configuration option 3-4
Default type changes 2-9
Defaults 1-9
delete 1-9
Display format changes 2-8
distinct
  in subqueries 2-23
  in views 1-10
double precision datatype 1-10
Dump commands
  and variables 1-6
dump database 1-5
  changes 2-6
Dump script changes 2-6
Dump striping 1-4
dump transaction 1-5
  changes 2-6
Duplicate rows 2-21
Dynamic SQL support 1-15

E

Equality comparisons 1-20
Error handling 1-23
Error logs 3-4
escape clause and **like** 1-17
exists predicate 2-22

F

fetch 1-9
FIPS Flagger 1-17
float datatype 1-10
Floating point conversions 2-11
for load option to **create database** 3-1
from clause 2-15
Functions
  Aggregate and column headings 1-15
  hextoint 1-14
  inttohex 1-14
  lct_admin 1-6
  overview 1-27
  rand 1-15

G

Global variables
  @@char_convert 3-6
  @@client_csname 3-6
  @@identity 1-11
  @@langid 3-3
  @@language 3-3
  @@maxcharlen 3-6
  @@ncharsize 3-5, 3-6
  @@transtate 1-13
grant command
  roles and 1-21
  in transactions 1-12
  with grant option 1-21
group by clause 1-15

H

Hex conversion functions 1-14, 1-27
hextoint function 1-14

I

Identifiers
  delimited 1-19
  finding reserved words 2-2
  quoted 1-19
Identifying users 1-8
identity property 1-11
in predicate 2-20
Index enhancements 1-23
index_col function 1-27
Inequality comparisons 1-20
Installing SQL Server 1-2
Integer conversions 2-11
Integrity constraints 1-9
Interfaces file 1-2
inttohex function 1-14, 1-27
Isolation levels 1-19
isql utility 1-21, 2-8

K

Keywords
  finding existing 2-2
  new 1-18, 2-2
kill command changes 1-22

L

langinstall utility program 3-2
language in cache configuration option 3-4
Languages
  adding 3-3
  aliases 3-3
  alternate 3-2
  dropping 3-3
  help 3-3
  installing 3-2
Last-chance threshold 1-6
lct_admin function 1-6, 1-27
Load commands
  and variables 1-6
load database 1-5
load transaction 1-5
Log segments 1-6
Login account locking 1-8

M

master database 1-2

Master device recovery 2-5
master..syslocks system table 1-33
max online engines configuration option 3-4
min online engines configuration option 3-4
Modifying SQL Server 1-2
money datatype 2-10
Monitoring space 1-5
Moving system procedures 1-3
Multibyte character sets 3-4
  length 3-6
Multi-file dumps 1-4

N

nchar datatype 3-5
Network dumps 1-4
no free space acctg
  sp_dboption 1-32
not in predicate 2-20
Null device 2-6
NULL operands 1-20
Numeric conversions 2-11
numeric datatype 1-10
Numeric literals 2-9
nvarchar datatype 3-5

O

object_name function 1-27
open 1-9
open databases configuration variable 2-3
Open databases, limit of 2-5
Operator role 1-7
Optimizer changes 1-24
Overflow conditions, conversion
    changes 2-10

P

Packet size, configuring 1-21, 1-24
Passwords
  aging 2-3
  encryption 1-8
  expiration 1-8
  length 1-8, 2-3
  null 2-3
patindex 3-6
Permissions
  changes to 1-20
  system procedures 2-3
Precompiler support 1-15
Primary key constraints 1-9
print arguments 3-5
privileges 1-18
proc_role function 1-27

Q

Query optimizer 1-24

R

raiserror 1-23
  arguments 3-5
rand function 1-15
real datatype 3-1
Recompiling stored procedures 3-3
Recovery changes, master device 2-5
Referential integrity constraints 1-9
Remote access 2-5
Repeated table names 2-15
reserved_pgs function 3-4
revoke command
  in transactions 1-12
  cascade option 1-21
  roles and 1-21
Roles 1-7, 2-14
rollback trigger 1-13
rowcnt function 3-4

S

Schemas 1-10
Security 1-6
Segments
  thresholds 1-5
select into 1-15, 2-15
Self-recursive triggers 1-13

set ansinull 1-20
set arithabort 1-18
set arithignore 1-18
set chained mode 1-19
set char_convert 3-5
set datefirst 3-2
set dateformat 3-2
set dup_in_subquery 2-19
set fipsflag 1-17
set isolation level 1-19
set language 3-2
set options overview 1-25
set quoted identifier 1-20
set self_recursion 1-13
set showplan 1-26
set string_rtruncation 1-16, 1-20
show_role function 1-27
shutdown command changes 1-22
Size limits for objects 1-13
smalldatetime datatype 3-1
smallmoney datatype 2-10, 3-1
some 1-18
Sort order 1-24
  help 3-3
Sort order changes
  and indexes 3-6
sp_addauditrecord 1-28
sp_addlanguage 3-3
sp_addlogin 3-3
sp_addmessage 3-6
sp_addsegment 1-30
sp_addthreshold 1-5, 1-28
sp_auditdatabase 1-28
sp_auditlogin 1-28
sp_auditobject 1-28
sp_auditoption 1-28
sp_auditsproc 1-28
sp_bindmsg 1-28
sp_checknames 3-3
sp_checkreswords 1-28, 2-2
sp_clearstats 1-21, 1-29
sp_configure 1-21
  additional netmem 1-30
  audit queue size 1-30
  cpu flush 1-21, 1-30
  default language 3-3, 3-4
  default network packet size 1-30
  default sortorder 3-4
  default sortorder id 3-3
  extent i/o buffers 1-23, 1-30
  i/o flush 1-21, 1-31
  identity burning set factor 1-31
  language in cache 3-3, 3-4
  max online engines 3-4
  maximum network packet size 1-31
  min online engines 3-4
  password expiration interval 1-31
  password expiration level 2-3
  remote access 2-5
  stack size 3-8
sp_configurelogin 1-28
sp_cursorinfo 1-9, 1-28
sp_dboption 1-31
  ddl in tran 1-12
  and thresholds 1-6
sp_dbremap 1-28
sp_defaultdb 1-30
sp_defaultlanguage 1-30, 3-3
sp_displaylogin 1-28
sp_dropmessage 3-6
sp_dropsegment 1-30
sp_dropthreshold 1-5, 1-28
sp_estspace 1-28
sp_extendsegment 1-30
sp_getmessage 3-6
sp_helpconstraint 1-9, 1-28
sp_helpdb 1-23, 1-30
sp_helpsegment 1-23, 1-30
sp_helpsort 3-3
sp_helpthreshold 1-5, 1-28
sp_indsuspect 3-6
sp_lock 1-30
sp_locklogin 1-29
sp_modifylogin 1-29
sp_modifythreshold 1-5, 1-29
sp_password 2-3
sp_procxmode 1-29, 2-14
sp_recompile 3-3

sp_remap 1-29
sp_reportstats 1-21, 1-29
sp_role 1-29
sp_setlangalias 3-3
sp_spaceused 3-4
sp_syntax 3-7
sp_thresholdaction 1-6, 1-29
sp_unbindmsg 1-29
sp_volchanged 1-4, 1-29
sp_who 1-30
  changes 1-22
Space available 1-23
Space monitoring 1-5
Spanish sort order 1-24
stack size configuration variable 3-8
Stored procedures
  moving 2-4
  recompiling 3-3
  size limits 1-13
  thresholds 1-5
Subqueries 1-14, 2-17 to 2-23
  and correlation names 2-17
Suspect indexes 3-6
syb_identity keyword 1-11
sybinit 1-2
sybsecurity database 1-8, 1-32
sybsystemprocs database 1-2, 2-3
Symmetric multiprocessor features 3-3
sysauditoptions system table 1-32
sysaudits system table 1-8, 1-32
syscharsets system table 3-2
syscolumns system table 1-33
syscomments system table 1-13, 1-33
sysconfigures system table 3-3
sysdatabases system table 1-33
sysengines system table 3-3
sysindexes system table 1-33, 3-2, 3-3
syslanguages system table 3-2
syslocks system table 3-3
syslogins system table 1-8, 1-33, 3-2
sysmessages system table 1-33, 3-2
sysobjects system table 1-33
sysprocesses system table 1-33, 3-3
sysprotects system table 1-33
System Administrator role 1-7, 2-14
System databases 1-2 to 1-3
System procedures 1-2
  sp_addauditrecord 1-28
  sp_addlanguage 3-3
  sp_addsegment 1-30
  sp_addthreshold 1-5, 1-28
  sp_auditdatabase 1-28
  sp_auditlogin 1-28
  sp_auditobject 1-28
  sp_auditoption 1-28
  sp_auditsproc 1-28
  sp_bindmsg 1-28
  sp_checknames 3-3
  sp_checkreswords 1-28, 2-2
  sp_clearstats 1-21, 1-29
  sp_configurelogin 1-28
  sp_cursorinfo 1-9, 1-28
  sp_dbremap 1-28
  sp_defaultdb 1-30
  sp_defaultlanguage 1-30, 3-3
  sp_displaylogin 1-28
  sp_droplanguage 3-3
  sp_dropsegment 1-30
  sp_dropthreshold 1-5, 1-28
  sp_estspace 1-28
  sp_extendsegment 1-30
  sp_helpconstraint 1-9, 1-28
  sp_helpdb 1-30
  sp_helplanguage 3-3
  sp_helpsegment 1-30
  sp_helpsort 3-3
  sp_helpthreshold 1-5, 1-28
  sp_lock 1-30
  sp_locklogin 1-29
  sp_modifylogin 1-29
  sp_modifythreshold 1-29
  sp_password 2-3
  sp_procxmode 1-29, 2-14
  sp_recompile 3-3
  sp_remap 1-29
  sp_reportstats 1-21, 1-29
  sp_role 1-29
  sp_setlangalias 3-3

  sp_thresholdaction 1-6, 1-29
  sp_unbindmsg 1-29
  sp_volchanged 1-4, 1-29
  sp_who 1-30
System Security Officer role 1-7
System tables 1-32, 1-33
systypes system table 1-33
sysusages system table 1-33
sysusermessages system table 3-5

T

Tables
  IDENTITY columns 1-11
Tapes
  and Backup Server 2-6
  labelling and handling 1-4
Threshold Manager 1-5
Transaction state information 1-13
Transactions
  data definition language and 1-12
  isolation levels and 1-19
  modes 1-19
Transact-SQL extensions 1-17
Triggers
  recompiling 3-3
  and **rollback trigger** 1-13
  self-recursion 1-13
Truncation
  character string 1-15, 1-20
  conversion changes 2-11

U

union operator 3-1
Unique constraints 1-9
update 1-9, 2-15
update statistics 1-24
Upgrading SQL Server 1-2
used_pgs function 3-4
user function 1-27
User-defined messages 3-6

V

valid_name 3-6
Variables, configuration. *See* **sp_configure**
Views 1-10

W

with check option clause 1-10
with grant option to **grant** command 1-21
work 1-18