SQL Server Reference Manual

Volume 2: System Procedures and Catalog Stored Procedures

SYBASE SQL Server Release 10.0

Document ID: 32402-01-1000-03

Change Level: 2

Last Revised: June 17, 1994

Principal authorship: Server Publications Group

Document ID: 32402-01-1000

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.

Sybase Trademarks

Restricted Rights Legend

Table of Contents

List of Tables

Preface

The *SQL Server Reference Manual* is a two-volume guide to SYBASE SQL Server™ and the Transact-SQL® language. This volume includes information about system procedures and catalog stored procedures. Volume 1, *Commands, Functions, and Topics* contains information about Transact-SQL commands, built-in functions, and topics of general interest to Transact-SQL users.

Audience

This manual is intended as a reference tool for Transact-SQL users of all levels. It provides basic syntax and usage information for every command, function, system procedure, and catalog stored procedure. It does not explain how to use these elements to build an application.

How to Use This Book

This manual consists of the following:

- Chapter 1, "System Procedures," contains reference pages for SQL Server system procedures.
- Chapter 2, "Catalog Stored Procedures," contains reference pages for SQL Server catalog stored procedures.
- Appendix A, "Reserved Words" lists Transact-SQL, APT-SQL, and SQL2 reserved words.
- Appendix B, "The System Tables" lists each system table and the definition of all columns in each table.
- Appendix C, "The pubs2 Database" describes the *pubs2* database in detail. It also includes an entity relationship diagram of all the tables in *pubs2*.
- The Index contains entries for both Volume 1 and Volume 2 of the *SQL Server Reference Manual*.
- Also included is an entity relationship diagram of the system tables.

Related Documents

Other manuals that you may find useful are:

- *What's New in SYBASE SQL Server Release 10.0*, which describes the new features software and documentation features of this release.
- *Transact-SQL User's Guide*, which documents Transact-SQL, Sybase's enhanced version of the SQL relational database language. It serves as a textbook for beginning users of the SYBASE database management system.
- *System Administration Guide*, which contains in-depth information about SQL Server system administration issues.
- *System Administration Guide Supplement*, which documents operating-system specific system administration tasks.
- *Open Client DB-Library Reference Manual*, a collection of manual pages and code samples for the SQL Server interface library, Open Client DB-Library.
- The *SQL Server Installation Guide,* which describes the installation procedures for SQL Server.
- *SQL Server Utility Programs* for your operating system, which describes utility programs that are invoked directly from the operating system.
- *Master Index for Server Publications* combines the indexes of the *SQL Server Reference Manual*, *Transact-SQL User's Guide*, and *System Administration Guide*. Use it to locate various topics in different contexts throughout the SYBASE documentation.

Conventions Used in This Manual

Formatting SQL Statements

SQL is a free-form language: there are no rules about the number of words you can put on a line, or where you must break a line. However, for readability, all examples and syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented.

SQL Syntax Conventions

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

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

Table 1: Syntax Statement Conventions

- Syntax statements (displaying the syntax and all options for a command) are printed like this:

  ```
  sp_dropdevice [device_name]
  ```

  or, for a command with more options:

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

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

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

  ```
  select * from publishers
  ```

- Examples of output from the computer are printed like this:

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

(3 rows affected)
```

Case

You can disregard case when you type keywords:

`SELECT` is the same as **`Select`** is the same as **`select`**

SQL Server's sensitivity to the case (upper or lower) of database objects (such as table names) and data depends on the sort order installed on your Server. Case sensitivity can be changed for single-byte character sets by reconfiguring SQL Server's sort order. (See the *System Administration Guide* for more information).

Obligatory Options {You Must Choose At Least One}

- **Curly Braces and Vertical Bars**: Choose **one and only one** option.

  ```
  {die_on_your_feet | live_on_your_knees |
  live_on_your_feet}
  ```

- **Curly Braces and Commas**: Choose one or more options. If you choose more than one, separate your choices with commas.

  ```
  {cash, check, credit}
  ```

Optional Options [You Don't Have to Choose Any]

- **One Item in Square Brackets**: You don't have to choose it.

  ```
  [anchovies]
  ```

- **Square Brackets and Vertical Bars:** Choose **none or only one**.

  ```
  [beans | rice | sweet_potatoes]
  ```

- **Square Brackets and Commas**: Choose **none, one, or more than one** option. If you choose more than one, separate your choices with commas.

  ```
  [extra_cheese, avocados, sour_cream]
  ```

Ellipsis: Do It Again (and Again)...

An ellipsis (three dots) means that you can **repeat** the last unit as many times as you like. In this syntax statement, **buy** is a required keyword:

```
buy thing = price [cash | check | credit]
    [, thing = price [cash | check | credit]]...
```

You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

Expressions

SQL Server syntax statements use several different types of expressions.

Usage	Definition
expression	Can include constants, literals, functions, column identifiers, variables or parameters
logical expression	An expression that returns TRUE, FALSE or UNKNOWN
constant expression	An expression that always returns the same value, such as “5+3” or “ABCDE”
float_expr	Any floating-point expression or expression that implicitly converts to a floating value
integer_expr	Any integer expression, or an expression that implicitly converts to an integer value
numeric_expr	Any numeric expression that returns a single value
char_expr	Any expression that returns a single character-type value
binary_expression	An expression that returns a single *binary* or *varbinary* value

Table 2: Types of Expressions Used in Syntax Statements

If You Need Help

Help with your SYBASE software is available in the form of documentation and the Technical Support Center.

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

System Procedures

1 System Procedures

This chapter describes the system procedures, which are Sybase-supplied stored procedures used for getting reports from and updating system tables. The following table lists the system procedures discussed in this chapter.

Procedure	Description
sp_addalias	Allows a SQL Server user to be known in a database as another user.
sp_addauditrecord	Allows users to enter user-defined audit records (comments) into the audit trail.
sp_addgroup	Adds a group to a database. Groups are used as collective names in granting and revoking privileges.
sp_addlanguage	Defines the names of the months and days for an alternate language and its date format.
sp_addlogin	Adds a new user account to SQL Server.
sp_addmessage	Adds user-defined messages to *sysusermessages* for use by stored procedure **print** and **raiserror** calls and by **sp_bindmsg**.
sp_addremotelogin	Authorizes a new remote server user by adding an entry to *master.dbo.sysremotelogins*.
sp_addsegment	Defines a segment on a database device in the current database.
sp_addserver	Defines a remote server, or defines the name of the local server.
sp_addthreshold	Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, SQL Server executes the associated stored procedure.
sp_addtype	Creates a user-defined datatype.
sp_addumpdevice	Adds a dump device to SQL Server.
sp_adduser	Adds a new user to the current database.
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 SQL Server user’s attempts to access tables and views; audits the text of a user’s command batches; lists users on which auditing is enabled; gives the auditing status of a user; or displays the status of table, view, or command text auditing.

Table 1-1: System Procedures

Procedure	Description
sp_auditobject	Audits accesses to tables and views.
sp_auditoption	Enables or disables system-wide auditing and global audit options, or reports on the status of audit options.
sp_auditsproc	Audits the execution of stored procedures and triggers.
sp_bindefault	Binds a default to a column or user-defined datatype.
sp_bindmsg	Binds a user message to a referential integrity constraint or check constraint.
sp_bindrule	Binds a rule to a column or user-defined datatype.
sp_changegroup	Changes a user's group.
sp_checknames	Checks the current database for names that contain characters not in the 7-bit ASCII set.
sp_checkreswords	Detects and displays identifiers that are Transact-SQL reserved words. Checks server names, device names, database names, segment names, user-defined datatypes, object names, column names, user names, login names, and remote login names.
sp_clearstats	Initiates a new accounting period for all server users or for a specified user. Prints statistics for the previous period by executing **sp_reportstats**.
sp_commonkey	Defines a common key—columns that are frequently joined—between two tables or views.
sp_configure	Displays or changes configuration variables.
sp_dboption	Displays or changes database options.
sp_dbremap	Forces SQL Server to recognize changes made by **alter database**. Run this procedure only if instructed to do so by SQL Server messages.
sp_depends	Displays information about database object dependencies—the view(s), trigger(s), and procedure(s) that depend on a specified table or view, and the table(s) and view(s) that are depended on by the specified view, trigger, or procedure.
sp_diskdefault	Sets a database device's status to **defaulton** or **defaultoff**. This indicates whether or not a database device can be used for database storage if the user does not specify a database device or specifies **default** with the **create database** or **alter database** commands.
sp_displaylogin	Displays information about a login account.
sp_dropalias	Removes the alias user name identity established with **sp_addalias**.

Table 1-1: System Procedures (continued)

Procedure	Description
sp_dropdevice	Drops a SQL Server database device or dump device.
sp_dropgroup	Drops a group from a database.
sp_dropkey	Removes from the *syskeys* table a key that had been defined using **sp_primarykey**, **sp_foreignkey**, or **sp_commonkey**.
sp_droplanguage	Drops an alternate language from the server and removes its row from *master.dbo.syslanguages*.
sp_droplogin	Drops a SQL Server user login by deleting the user's entry in *master.dbo.syslogins*.
sp_dropmessage	Drops user-defined messages from *sysusermessages*.
sp_dropremotelogin	Drops a remote user login.
sp_dropsegment	Drops a segment from a database or unmaps a segment from a particular database device.
sp_dropserver	Drops a server from the list of known servers.
sp_dropthreshold	Removes a free-space threshold from a segment.
sp_droptype	Drops a user-defined datatype.
sp_dropuser	Drops a user from the current database.
sp_estspace	Estimates the amount of space required for a table and its indexes, and the time needed to create the index.
sp_extendsegment	Extends the range of a segment to another database device, or extends an existing segment on the current database device.
sp_foreignkey	Defines a foreign key on a table or view in the current database.
sp_getmessage	Retrieves stored message strings from *sysmessages* and *sysusermessages* for **print** and **raiserror** statements.
sp_help	Reports information about a database object (any object listed in *sysobjects*), and about SQL Server-supplied or user-defined datatypes.
sp_helpconstraint	Reports information about any integrity constraints specified for a table. This information includes the constraint name and the definition of the default, unique / primary key constraint, referential constraint, or check constraint.
sp_helpdb	Reports information about a particular database or about all databases.
sp_helpdevice	Reports information about a particular device or about all SQL Server database devices and dump devices.
sp_helpgroup	Reports information about a particular group or about all groups in the current database.

Table 1-1: System Procedures (continued)

Procedure	Description
sp_helpindex	Reports information about the indexes created on a table.
sp_helpjoins	Lists the columns in two tables or views that are likely join candidates.
sp_helpkey	Reports information about a primary, foreign, or common key of a particular table or view, or about all keys in the current database.
sp_helplanguage	Reports information about a particular alternate language or about all languages.
sp_helplog	Reports the name of the device that contains the first page of the log.
sp_helpremotelogin	Reports information about a particular remote server's logins or about all remote servers' logins.
sp_helprotect	Reports on permissions for database objects, users, or groups.
sp_helpsegment	Reports information on a particular segment or on all of the segments in the current database.
sp_helpserver	Reports information about a particular remote server or about all remote servers.
sp_helpsort	Displays SQL Server's default sort order and character set.
sp_helptext	Prints the text of a system procedure, trigger, view, default, rule, or integrity check constraint.
sp_helpthreshold	Reports the segment, free-space value, status, and stored procedure associated with all thresholds in the current database or all thresholds for a particular segment.
sp_helpuser	Reports information about a particular user or about all users in the current database.
sp_indsuspect	Checks user tables for indexes that have been marked as suspect during recovery following a sort order change.
sp_lock	Reports information about processes that currently hold locks.
sp_locklogin	Locks a SQL Server account so that the user cannot log in, or displays a list of all locked accounts.
sp_logdevice	Puts the system table *syslogs*, which contains the transaction log, on a separate database device.
sp_modifylogin	Modifies the default database, default language, or full name for a SQL Server login account.

Table 1-1: System Procedures (continued)

Procedure	Description
sp_modifythreshold	Modifies a threshold by associating it with a different threshold procedure, level of free space, or segment. You **cannot** use **sp_modifythreshold** to change the amount of free space or the segment name for the last-chance threshold.
sp_monitor	Displays statistics about SQL Server.
sp_password	Adds or changes a password for a SQL Server login account.
sp_placeobject	Puts future space allocations for a table or index on a particular segment.
sp_primarykey	Defines a primary key on a table or view.
sp_procxmode	Displays or changes the transaction modes associated with stored procedures.
sp_recompile	Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.
sp_remap	Remaps a Release 4.8 or later stored procedure, trigger, rule, default, or view to be compatible with Release 10.0. Use **sp_remap** on objects that the Release 10.0 upgrade procedure failed to remap.
sp_remoteoption	Displays or changes remote login options.
sp_rename	Changes the name of a user-created object in the current database.
sp_renamedb	Changes the name of a database. You **cannot** rename system databases or databases with external referential integrity constraints.
sp_reportstats	Reports statistics on system usage.
sp_role	Grants or revokes roles to a SQL Server login account.
sp_serveroption	Displays or changes remote server options.
sp_setlangalias	Assigns or changes the alias for an alternate language.
sp_spaceused	Displays the number of rows, the number of data pages, and the space used by one table or by all tables in the current database.
sp_syntax	Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines (depending on which products and corresponding **sp_syntax** scripts exist on your server).
sp_unbindefault	Unbinds a created default value from a column or from a user-defined datatype.
sp_unbindmsg	Unbinds a user-defined message from a constraint.

Table 1-1: System Procedures (continued)

Procedure	Description
sp_volchanged	Notifies the Backup Server that the operator performed the requested volume handling during a dump or load. (OpenVMS users can use REPLY instead.)
sp_who	Reports information about all current SQL Server users and processes, or about a particular user or process.

Table 1-1: System Procedures (continued)

Introduction to System Procedures

The system procedures, created by **installmaster** at installation, are located in the *sybsystemprocs* database and are owned by the System Administrator, but many of them can be run from any database.

If a system procedure is executed in a database other than *sybsystemprocs*, it operates on the system tables in the database from which it was executed. For example, if the Database Owner of *pubs2* runs **sp_adduser** from *pubs2*, the new user is added to *pubs2..sysusers*.

Permissions on System Procedures

Since system procedures are located in the *sybsystemprocs* database, their permissions are also set there.

Some system procedures can be run only by Database Owners. These procedures make sure that the user executing the procedure is the owner of the database from which they are being executed.

Other system procedures (for example, all the **sp_help** procedures) can be executed by any user who has been granted permission—but this permission must be granted in *sybsystemprocs*. In other words, a user must have permission to execute a system procedure in all databases, or in none of them.

Users not listed in *sybsystemprocs..sysusers* are treated as "guest" in *sybsystemprocs*, and thus are automatically granted permission on many of the system procedures. To deny a user permission on a system procedure, the System Administrator must add the user to *sybsystemprocs..sysusers* and write a **revoke** statement that applies to that procedure. The owner of a user database cannot directly control permissions on the system procedures within his or her own database.

Using System Procedures

If a parameter value for a system procedure contains punctuation or embedded blanks, or is a reserved word, you must enclose it in single or double quotes. If the parameter is an object name qualified by a database name or owner name, enclose the entire name in single or double quotes.

➤ ***Note***

Do not use delimited identifiers as system procedure parameters; they may produce unexpected results.

All system procedures report a return status. For example:

```
return status = 0
```

means that the procedure executed successfully. The examples in this book do not include the return status.

You can create your own system procedures that can be executed from any database. (See Chapter 1, ''Overview of System Administration'' in the *System Administration Guide* for more information.)

Values for Optional Parameters

If a procedure has multiple optional parameters you can supply parameters in the form:

```
@parametername = value
```

than to supply all of the parameters. The parameter names in the syntax statements match the parameter names defined by the procedures.

For example, the syntax for **sp_addlogin** is:

```
sp_addlogin login_name, password [, defdb
    [, deflanguage [, fullname]]]
```

To use **sp_addlogin** to create a login for “susan” with a password of “wonderful”, a full name of Susan B. Anthony, and the server’s default database and language, you can use:

```
sp_addlogin susan, wonderful,
        @fullname="Susan B. Anthony"
```

This provides the same information as the command with all of the parameters specified:

```
sp_addlogin susan, wonderful, public_db,
    us_english, "Susan B. Anthony"
```

You can also use “null” as a placeholder:

```
sp_addlogin susan, wonderful, null, null,
        "Susan B. Anthony"
```

Do not enclose “null” in quotes.

System Procedure Tables

Several **system procedure tables** in the *master* database are used by system procedures to convert internal system values (for example, status bits) into human-readable format. One of them, *spt_values*, is used by **sp_addsegment**, **sp_addtype**, **sp_addumpdevice**, **sp_checkreswords**, **sp_commonkey**, **sp_configure**, **sp_dboption**, **sp_depends**, **sp_dropsegment**, **sp_dropuser**, **sp_estspace**, **sp_extendsegment**, **sp_foreignkey**, **sp_help**, **sp_helpdb**, **sp_helpdevice**, **sp_helpindex**, **sp_helpjoins**, **sp_helpkey**, **sp_helplog**, **sp_helpremotelogin**, **sp_helprotect**, **sp_helpsegment**, **sp_helpserver**, **sp_helpsort**, **sp_remoteoption**, **sp_renamedb**, **sp_serveroption**, **sp_setreplicate**, and **sp_spaceused**.

spt_values is never updated. To see how it is used, execute **sp_helptext** to look at the text for one of the system procedures that references it.

The other system procedure tables are *spt_committab* and *spt_monitor*. In addition, some system procedures create and then drop temporary tables: **sp_helpdb** creates *#spdbdesc*; **sp_helpdevice** creates *#spdevtab*; **sp_helpindex** creates *#spindtab*; and **sp_helpserver** creates *#spt_server*.

sp_addalias

Function

Allows a SQL Server user to be known in a database as another user.

Syntax

`sp_addalias` ***`login_name`*****`,`** ***`name_in_db`***

Parameters

login_name – is the *master.dbo.syslogins* name of the user who wants an alternate identity in the current database.

name_in_db – is the database user name to alias *login_name* to. The name must exist in both *master.dbo.syslogins* and in the *sysusers* table of the current database.

Examples

1. **`sp_addalias victoria, albert`**

   There is a user named "albert" in the database's *sysusers* table and a login for a user named "victoria" in *master.dbo.syslogins*. This command allows "victoria" to use the current database by assuming the name "albert".

Comments

- Executing **sp_addalias** maps one user to another in the current database. The mapping is shown in *sysalternates*, where the two users' *suids* are connected.
- A user may be aliased to only one database user at a time.
- A report on any users mapped to a specified user can be generated with **sp_helpuser**, giving the specified user's name as an argument.
- When a user tries to use a database, SQL Server checks *sysusers* to see if the user is listed there. If the user is not there, it then checks *sysalternates*. If the user's *suid* is in *sysalternates*, mapped to a database user's *suid*, the first user is treated as the second user while using the database.

  If the user named in *login_name* is in the database's *sysusers* table, SQL Server won't find the user's alias identity, since it checks *sysusers* before checking *sysalternates*, where the alias is listed.

Messages

- `Alias user added.`

  The procedure was successful. Now *login_name* can use the current database. While doing so, the user is known as *name_in_db*.

- `'`*`login_name`*`' is already a user in the current database.`

  A user with a login in the current database cannot be aliased to another login in that database.

- `No login with the specified name exists.`

  There is no entry in *master.dbo.syslogins* for *login_name*. Everyone using SQL Server, whether aliased or not, must have a login.

- `No user with the specified name exists in the current database.`

  Since *name_in_db* is not a user in the database, *login_name* can't be aliased to it.

- `The specified user name is already aliased.`

  The *login_name* is already aliased to a user in the current database. A *login_name* may be aliased to only one database user at a time. To change an alias, first drop the current alias using **sp_dropalias**. Then add the new alias.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addalias**.

Tables Used

master.dbo.syslogins, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**use**
System procedures	**sp_addlogin, sp_adduser, sp_dropalias, sp_helpuser**

sp_addauditrecord

Function

Allows users to enter user-defined audit records (comments) into the audit trail.

Syntax

```
sp_addauditrecord [@text="message text"]
   [, @db_name="db_name"] [, @obj_name="object_name"]
   [, @owner_name="object_owner"]
   [, @dbid=database_ID] [, @objid=object_ID]
```

Parameters

@text – is the text of the message to add to *sysaudits*. The text is inserted into the *extrainfo* field of *sysaudits*.

@db_name – is the name of the database referred to in the record. This is inserted into the *dbname* field of *sysaudits*.

@obj_name – is the name of the object referred to in the record. This is inserted into the *objname* field of *sysaudits*.

@owner_name – is the owner of the object referred to in the record. This is inserted into the *objowner* field of *sysaudits*.

@dbid – is the database ID number of *db_name*. Do not enclose this integer value in quotes. **@dbid** is inserted into the *dbid* field of *sysaudits*.

@objid – is the object ID number of *obj_name*. Do not enclose this integer value in quotes. **@objid** is inserted into the *objid* field of *sysaudits*.

Examples

1.
   ```
   sp_addauditrecord @text="I gave A. Smith
   permission to view the payroll table in the
   corporate database. This permission was in effect
   from 3:10 to 3:30 pm on 9/22/92.",
   @db_name="corporate", @obj_name="payroll",
   @owner_name="dbo", @dbid=10, @objid=1004738270
   ```

   Adds this record to *sysaudits*. The message portion is entered into the *extrainfo* field of *sysaudits*, "corporate" into the *dbname* field, "payroll" into the *objname* field, "dbo" into the *objowner* field, "10" into the *dbid* field, and "1004738270" into the *objid* field.

2. **`sp_addauditrecord @text="I am disabling auditing briefly while we reconfigure the system", @db_name="corporate"`**

   Adds this record to *sysaudits*. This example uses parameter names, with the @ prefix. This format allows you to leave some fields empty.

Comments

- You can use **sp_addauditrecord** if:
  - You have been granted execute permission on **sp_addauditrecord**. (No special role is required.)
  - Auditing is enabled (**sp_auditoption "enable auditing"** is set to **on**).
  - The **adhoc records** option of **sp_auditoption** is set to **on.**
- **sp_addauditrecord** does not check the correctness of the information you enter. For example, it does not check to see if the database ID you have entered is correct for the database referred to in the audit record.

Messages

None.

Permissions

Permission to execute **sp_addauditrecord** defaults to the System Security Officer and the Database Owner of the *sybsecurity* database. The Database Owner can grant execute permission to other users.

Tables Used

sybsecurity.dbo.sysaudits

See Also

Topics	Auditing
System procedures	**sp_auditoption**

sp_addgroup

Function

Adds a group to a database. Groups are used as collective names in granting and revoking privileges.

Syntax

```
sp_addgroup grpname
```

Parameters

grpname – is the name of the group. Group names must conform to the rules for identifiers.

Examples

1. **`sp_addgroup accounting`**

   Creates a group named *accounting*.

Comments

- **sp_addgroup** adds the new group to a database's *sysusers* table. Each group's user ID (*uid*) is 16384 or larger (except "public," which is always 0).
- Once a group has been created, add new users with **sp_adduser**. To add an already existing user to a group, use **sp_changegroup**.
- Every database is created with a group named "public." Every user is automatically a member of "public." Each user can be a member of one additional group.

Messages

- `A group with the specified name already exists.`

  The group name you supplied is being used as a group name. Choose another name.
- `A user with the specified group name already exists.`

  The group name you supplied is being used as a user name. Choose another name.
- *`grpname`* `is not a valid name.`

  Group names must conform to the rules for identifiers.

- `New group added.`

  The group has been added to the current database's *sysusers* table.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addgroup**.

Tables Used

sysobjects, *sysusers*

See Also

Commands	**grant, revoke**
System procedures	**sp_adduser, sp_changegroup, sp_dropgroup, sp_helpgroup**

sp_addlanguage

Function

Defines the names of the months and days for an alternate language, and its date format.

Syntax

```
sp_addlanguage language, alias, months, shortmons,
   days, datefmt, datefirst
```

Parameters

language – is the official language name for the language, entered in 7-bit ASCII characters only.

alias – substitutes for the alternate language's official name. Enter either "null" to make the alias the same as the official language name, or a name you prefer. You can use 8-bit ASCII characters in an alias—française, for example—if your terminal supports them.

months – is a list of the full names of the 12 months, ordered from January through December, separated only by commas (no spaces allowed). Month names can be up to 20 characters long and can contain 8-bit ASCII characters.

shortmons – is a list of the abbreviated names of the 12 months, ordered from January through December, separated only by commas (no spaces allowed). Month abbreviations can be up to nine characters long and can contain 8-bit ASCII characters.

days – is a list of the full names of the seven days, ordered from Monday through Sunday, separated only by commas (no spaces allowed). Day names can be up to 30 characters long and can contain 8-bit ASCII characters.

datefmt – is the date order of the date parts *month/day/year* for entering *datetime* or *smalldatetime* data. Valid arguments are *mdy*, *dmy*, *ymd*, *ydm*, *myd*, or *dym*. "dmy" indicates that dates are in day / month / year order.

datefirst – sets the number of the first weekday for date calculations. For example, Monday is 1, Tuesday is 2, and so on.

Examples

1.
```
sp_addlanguage french, null,
    "janvier,fevrier,mars,avril,mai,juin,juillet,
    aout,septembre,octobre,novembre,decembre",
    "jan,fev,mars,avr,mai,juin,jui,aout,sept,oct,
    nov,dec",
    "lundi,mardi,mercredi,jeudi,vendredi,samedi,
    dimanche",
    dmy, 1
```

This stored procedure adds French to the languages available on the server. “null” makes the alias the same as the official name, “french”. Date order is “dmy”—”day / month / year”. “1” specifies that lundi, the first item in the *days* list, is the first weekday. Because the French do not capitalize the names of the days and months except when they appear at the beginning of a sentence, this example shows them being added in lowercase.

Comments

- Normally, add alternate languages from one of SQL Server’s Language Modules using the **sybinit** installation program. A Language Module supplies the names of the dates and translated error messages for that language. However, if a Language Module is not provided with your server, use **sp_addlanguage** to define the date names and format.
- **sp_addlanguage** creates an entry in *master.dbo.syslanguages*, inserting a unique numeric value in the *langid* column for each alternate language. *langid* 0 is reserved for us_english.
- Users can display a list of the alternate languages on the server with **sp_helplanguage**. They can change their own default language to any on the list with **sp_modifylogin**.
- The official language name in the *name* column of *master.dbo.syslanguages* must be unique.
- The *alias* column in *master.dbo.syslanguages* is set to the official language name if NULL is entered for *alias*, but System Administrators can change the value of *syslanguage.alias* with **sp_setlangalias**.
- The *upgrade* column in *master.dbo.syslanguages* is set to 0.
- SQL Server sends date values to clients as *datatime* datatype, and the clients use localization files to display the dates in the user’s current language. For date strings added with **sp_addlanguage**, use

the **convert** function to convert the dates to character data in the server:

```
select convert(char, pubdate) from table
```

where *pubdate* is *datetime* data and *table* is any table.

- When users perform data entry on date values and need to use date names created with **sp_addlanguage**, the client must have these values input as character data, and sent to the server as character data.
- If users set default languages to a language added with **sp_addlanguage**, and there are no localization files for the language, they receive an informational message when they log in, indicating that their client software could not open the localization files.

Messages

- `'language' already exists in syslanguages.`

  This language already exists on the server. To change only the language alias, use **sp_setlangalias**. To change *months*, *shortmons*, *days*, *datefmt*, or *datefirst*, drop the language with **sp_droplanguage**, then add it again with your new specifications.

- `List of full month names contains spaces, which are not allowed.`

  Separate month names only by commas; no spaces are allowed.

- `List of full month names contains name(s) which have iso_1 non-alphabetic characters.`

  Month names cannot contain non-alphabetic characters, such as punctuation.

- `List of full month names has too few names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of full month names has too many names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of full month names has name(s) which are too long.`

  One or more names in the list of full month names is more than 20 characters long.

- `List of short month names contains spaces, which are not allowed.`

  Short month names cannot contain non-alphabetic characters, such as spaces.

- `List of short month names contains name(s) which have iso_1 non-alphabetic characters.`

  Short month names cannot contain non-alphabetic characters, such as punctuation.

- `List of short month names has too few names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of short month names has too many names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of short month names has name(s) which are too long.`

  One or more names in the list of short month names is more than nine characters long.

- `List of day names contains spaces, which are not allowed.`

  Day names cannot contain non-alphabetic characters, such as spaces.

- `List of day names contains name(s) which have iso_1 non-alphabetic characters.`

  Day names cannot contain non-alphabetic characters, such as punctuation.

- `List of day names has too few names.`

  The days list must have exactly seven names separated by exactly six commas.

- `List of day names has too many names.`

  The days list must have exactly seven names separated by exactly six commas.

- `List of day names has name(s) which are too long.`

  One or more names in the list of day names is more than 30 characters long.

- `'`*`datefmt`*`' is not a valid date order.`

  datefmt must be in one of the following six orders: “mdy”, “myd”, “dmy,” “dym”, “ydm”, “ymd”.

- `'`*`datefirst`*`' is not a valid first day.`

  The first day of a week must be 1 for Monday through 7 for Sunday.

- `'`*`alias`*`' alias already exists in syslanguages.`

  The name given as an alias is already in use as an alias in the table *master.dbo.syslanguages*. If *alias* was specified as NULL, then the official language name for this new language is already in use as the alias for another language.

- `Language not inserted.`

  An error occurred while adding this language to *master.dbo.syslanguages*, so the language was not added. The SQL Server error message that appeared before this message gives more specific information about the error.

- `New language inserted.`

  A new alternate language was added to SQL Server and *master.dbo.syslanguages*.

Permissions

Only a System Administrator can execute **sp_addlanguage**.

Tables Used

master.dbo.syslanguages, *sysobjects*

See Also

Commands	**set**
System procedures	**sp_droplanguage, sp_helplanguage, sp_setlangalias, sp_modifylogin**

sp_addlogin

Function

Adds a new user account to SQL Server.

Syntax

```
sp_addlogin login_name, password [, defdb
    [, deflanguage [, fullname]]]
```

Parameters

login_name – is the user's login name. Login names must conform to the rules for identifiers. It is highly recommended that users' SQL Server login names be the same as their operating system login names. This makes login to SQL Server easier, simplifies management of server and operating system login accounts, and makes it easier to correlate audit data generated by SQL Server and by the operating system.

password – is the user's password. Passwords must be at least six bytes long. If you specify a shorter password, **sp_addlogin** returns an error message and exits.

defdb – is the name of the default database assigned when a user logs in to SQL Server. If you do not specify *defdb*, the default is *master*.

deflanguage – is the official name of the default language assigned when a user logs in to SQL Server. The server's default language, defined by the configuration variable **default language**, is used if you do not specify *deflanguage*.

fullname – is the full name of the user who owns the login account. This can be used for documentation and identification purposes.

Examples

1. **`sp_addlogin albert, longer1, corporate`**

   Creates a SQL Server login for "albert". His password is "longer1" and his default database is *corporate*.

2. **`sp_addlogin claire, bleurouge, public_db, french`**

   Creates a SQL Server login for "claire". Her password is "bleurouge", her default database is *public_db*, and her default language is French.

3. **`sp_addlogin robertw, terrible2, public_db, null,`**
   `"Robert Willis"`

   Creates a SQL Server login for “robertw”. His password is “terrible2”, his default database is *public_db*, and his full name is “Robert Willis”. **null** must not be placed in quotes.

4. **`sp_addlogin susan, wonderful, null, null,`**
   `"Susan B. Anthony"`

   Creates a login for “susan” with a password of “wonderful”, a full name of Susan B. Anthony, and the server’s default database and language. **null** must not be placed in quotes.

5. **`sp_addlogin susan, wonderful,`**
   `@fullname="Susan B. Anthony"`

   An alternative way of creating Example 4, using the parameter name “@fullname”.

Comments

- For ease of management, it is highly recommended that all users’ SQL Server login names be the same as their operating system login names.
- After assigning a default database to a user with **sp_addlogin**, the Database Owner or System Administrator must provide access to the database by executing **sp_adduser** or **sp_addalias**.
- Although a user can use **sp_modifylogin** to change his or her own default database at any time, a database cannot be used without permission from the Database Owner.
- A user can use **sp_password** at any time to change his or her own password. A System Security Officer can use **sp_password** to change any user’s password.
- A user can use **sp_modifylogin** to change his or her own default language. A System Administrator can use **sp_modifylogin** to change any user’s default language.
- A user can use **sp_modifylogin** to change his or her own *fullname*. A System Administrator can use **sp_modifylogin** to change any user’s *fullname*.

Messages

- `'`*`login_name`*`' is not a valid name.`

  login_name must conform to the rules for identifiers. See “Identifiers” in Volume 1 of the *SQL Server Reference Manual*.

- `'`*`deflanguage`*`' is not an official language name from syslanguages.`

  Use **sp_helplanguage** to determine the alternate languages available. Add an alternate language with **langinstall**, or specify us_english.

- `Can't run sp_addlogin from within a transaction.`

  sp_addlogin modifies system tables, so it cannot be run within a transaction.

- `A user with the specified login name already exists.`

  Choose another *login_name*. If you only want to change the user's password, default database, or default language, use **sp_password** or **sp_modifylogin**.

- `Database name not valid -- login not added.`

  The specified default database does not exist. Create the database first or choose a database that already exists.

- `New login created.`

Permissions

Only a System Administrator can execute **sp_addlogin**.

Tables Used

master.dbo.sysdatabases, master.dbo.syslogins, sysobjects

See Also

Topics	Login Management, Roles
System procedures	**sp_addalias, sp_adduser, sp_auditsproc, sp_droplogin, sp_locklogin, sp_modifylogin, sp_password, sp_role**

sp_addmessage

Function

Adds user-defined messages to *sysusermessages* for use by stored procedure **print** and **raiserror** calls and by **sp_bindmsg**.

Syntax

sp_addmessage *message_num*, *message_text* [, *language*]

Parameters

message_num – is the message number of the message to add. The message number for a user-defined message must be 20000 or greater.

message_text –is the text of the message to add.The maximum length is 255 bytes. **print**, **raiserror**, and **sp_bindmsg** recognize place holders in the message text to print out. A single message can contain up to 20 unique place holders in any order. These place holders are replaced with the formatted contents of any arguments that follow the message when the text of the message is sent to the client.

The place holders are numbered to allow reordering of the arguments when translating a message to a language with a different grammatical structure. A place holder for an argument appears as "*%nn*!", a percent sign (%), followed by an integer from 1 to 20, followed by an exclamation point (!). The integer represents the argument number in the string in the argument list. "%1!" is the first argument in the original version, "%2!" is the second argument, and so on.

language – is the language of the message to add. This must be a valid language name in *syslanguages* table. If this parameter is missing, SQL Server assumes that messages are in the default session language indicated by *@@langid*.

Examples

1.
```
sp_addmessage 20001,
"The table '%1!' is not owned by the user '%2!'."
```
   Adds a message with the number 20001 to *sysusermessages*.

Comments

- **sp_addmessage** does not overwrite an existing message of the same number and *langid*. Drop the message using **sp_dropmessage** first.

Messages

- `'`*`language`*`' is not an official language name from syslanguages.`

  Use **sp_helplanguage** to see the list of official language names available on this SQL Server.

- `Message number must be at least 20000.`

  User-defined messages must have a message number of 20000 or greater.

- `Cannot add message until sysusermessages system table is created properly by Upgrade.`

  sysusermessages was added to SQL Server in Release 4.9. This SQL Server has not been properly upgraded to 4.9. See your installation guide for information on upgrade.

- `A message with number `*`message_number`*` in the specified language already exists. Drop the old message first if you still wish to add this one.`

  You attempted to insert a message with a number that already exists in *sysusermessages*.

- `The message has not been inserted.`

  sp_addmessage failed. *sysusermessages* is unchanged.

- `The message has been inserted.`

  You successfully added a message to *sysusermessages*.

Permissions

Any user can execute **sp_addmessage**.

Tables Used

master.dbo.syslanguages, *sysobjects*, *sysusermessages*

See Also

Commands	**print, raiserror**
System procedures	**sp_dropmessage, sp_getmessage**

sp_addremotelogin

Function

Authorizes a new remote server user by adding an entry to *master.dbo.sysremotelogins*.

Syntax

```
sp_addremotelogin remoteserver [, login_name
   [, remote_name] ]
```

Parameters

remoteserver – is the name of the remote server to which the remote login applies. This server must be known to the local server by an entry in the *master.dbo.sysservers* table, created with **sp_addserver**.

login_name – is the login name of the user on the local server. *login_name* must already exist in the *master.dbo.syslogins* table.

remote_name – is the name that the remote server uses when logging into the local server. All *remote_name*s that aren't explicitly matched to a local *login_name* are automatically matched to a local name. In example 1, the local name is the remote name used to log in. In example 2, the local name is "albert".

Examples

1. **`sp_addremotelogin GATEWAY`**

   This creates an entry in the *sysremotelogins* table for the remote server GATEWAY, for purposes of login validation. This is a simple way to map remote names into local names when the local and remote servers have the same users.

2. **`sp_addremotelogin GATEWAY, albert`**

   This creates an entry that maps the remote server GATEWAY to a local user name "albert".

3. **`sp_addremotelogin GATEWAY, churchy, pogo`**

   This causes a remote login from the remote user "pogo" on the remote server GATEWAY to be mapped into the local user "churchy".

Comments

- When a remote login is received, the local server tries to map the remote user into a local user in three different ways:
  - First, the local server looks for an entry in *sysremotelogins* that matches the remote server name and the remote user name. If one is found, then the local server user ID for that row is used to log the remote user in.
  - If no entry is found, the local server searches for an entry that has a remote name of NULL and a local server user ID that is not -1. In this case, the remote user is mapped to the local server user ID.
  - Finally, if the previous attempts failed, the *sysremotelogins* table is checked for an entry that has a remote name of NULL and a local server user ID that is -1. In this case, whatever remote name was supplied by the remote server is used to look for a local server user ID in the *syslogins* table.
- The name of the local user may be different on the remote server.
- Every remote login entry has a status. The default status for the **trusted** option is "false" (that is, not trusted). This means that when a remote login comes in using that entry, the password is checked. If you don't want the password to be checked, change the status of the **trusted** option to "true" with **sp_remoteoption**.

Messages

- `'`*`login_name`*`' isn't a local user -- remote login denied.`

  The *login_name* isn't in the *master..syslogins* table. If you supply a local name, it must currently exist as a user on the local server.
- `New remote login created.`

  A remote login was created.
- `'`*`remoteserver`*`' is the local server - remote login not applicable.`

  You have tried to define a remote login to the local server. Logins to the local server are listed in *master.dbo.syslogins*.
- `There is already a default-name mapping of a remote login from remote server '`*`remoteserver`*`'.`

  You have tried to add a duplicate remote login entry. See Examples 1 and 2. Use **sp_helpremotelogins** to see the remote logins for the *remoteserver*.

- `There is already a remote user named '`*`remotename`*`' for remote server '`*`remoteserver`*`'.`

  A user with that remote login name for that remote server already exists. Drop that remote user before choosing another *remotename*.

- `There is not a server named '`*`server`*`'.`

  The specified remote server does not exist. Use **sp_helpserver** to get a list of the existing remote servers.

- `Usage:sp_addremotelogin remoteserver [, loginame [, remotename]]`

  Syntax summary. You have incorrectly specified a parameter to **sp_addremotelogin.**

- `Can’t run sp_addremotelogin from within a transaction.`

  sp_addremotelogin modifies system tables, so it cannot be run within a transaction.

Permissions

Only a System Administrator can execute **sp_addremotelogin**.

Tables Used

master.dbo.syslogins, master.dbo.sysremotelogins, master.dbo.sysservers, sysobjects

See Also

System procedures	**sp_addlogin, sp_addserver, sp_dropremotelogin, sp_helpremotelogin, sp_helpserver, sp_remoteoption**

sp_addsegment

Function

Defines a segment on a database device in the current database.

Syntax

sp_addsegment *segname*, *dbname*, *devname*

Parameters

segname – is the name of the new segment to add to the *syssegments* table of the database. Segment names are unique in each database.

dbname – specifies the name of the database on which to define the segment. *dbname* must be the name of the current database.

devname – is the name of the database device on which to locate *segname*. A database device can have more than one segment associated with it.

Examples

1. **sp_addsegment indexes, pubs2, dev1**

   This command creates a segment named *indexes* for the database *pubs2* on the database device named *dev1*.

Comments

- **sp_addsegment** defines segment names for database devices assigned to a specific database with an **alter database** or **create database** command.
- After defining a segment, use it in **create table** and **create index** commands and in the **sp_placeobject** procedure to place a table or index on the segment.

  When a table or index is created on a particular segment, all the subsequent data for the table or index is located on the segment.
- Use the system procedure **sp_extendsegment** to extend the range of a segment to another database device used by the same database.
- If a database is extended with **alter database** on a device used by that database, the segments mapped to that device are also extended.

- The *system* and *default* segments are mapped to each database device included in a **create database** or **alter database** command. The *logsegment* is also mapped to each device, unless it is placed on a separate device with the **log on** extension to **create database** or later use of **sp_logdevice**. Use **sp_dropsegment** to unmap these segments, if desired. See the *System Administration Guide* for more information.

Messages

- `Can’t run sp_addsegment from within a transaction.`

  sp_addsegment modifies system tables, so it cannot be run within a transaction.

- `'`*`devname`*`' is reserved exclusively as a log device.`

  You can’t create a segment on a disk device that is dedicated to the database log.

- `No such device exists -- run sp_helpdb to list the devices for the current database.`

  The named device doesn’t exist in *sysdevices*.

- `Segment created.`

  The procedure was successful; *segname* is now in the current database.

- `'`*`segname`*`' is not a valid identifier.`

  Segment names must conform to the rules for identifiers. They must begin with a letter, an underscore (_), or pound sign (#). After the first character, identifiers can include letters, underscores, pound signs, or dollar signs ($).

- `The maximum number of segments for the current database are already defined.`

  A database can have no more than 31 segments. You can drop a segment with **sp_dropsegment** and replace it with a new one.

- `The specified device is not used by the database.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not used by the specified database, and therefore, a segment cannot be added to it. Segments may only be defined on database devices used by the database. The **alter database** command can extend a database on a device listed in *master.dbo.sysdevices*.

- `The specified device is not a database device.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not a database device. It may be a dump device.

- `There is already a segment named '`*`segname`*`'.`

  Segment names must be unique in each database.

- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use `*`database_name`*`' and try again.`

  sp_addsegment can add segments only in the database you are currently using. Issue the **use** command to open the database in which you want to add a segment. Then run **sp_addsegment** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addsegment.**

Tables Used

master.dbo.sysdevices, *master.dbo.sysusages*, *sysobjects*, *syssegments*

See Also

Commands	**alter database, create index, create table, disk init**
System procedures	**sp_dropsegment, sp_extendsegment, sp_helpdb, sp_helpdevice, sp_placeobject**

sp_addserver

Function

Defines a remote server, or defines the name of the local server.

Syntax

```
sp_addserver srvname [, {local | null}
   [, network_name]]
```

Parameters

srvname – is the name by which to address the server on your system. **sp_addserver** adds a row to the *sysservers* table if there is no entry already present for *srvname*. Server names must be unique, and must conform to the rules for identifiers.

local | **null** – identifies the server being added as a local server. The **local** value is used only once after start-up, or after a reboot, to identify the local server name so that it can appear in messages printed by SQL Server. **null** specifies that this server is a remote server.

network_name – is the name in the *interfaces* file for the server named *srvname*. This enables you to establish local aliases for other SQL Servers or Backup Servers that you may need to communicate with. If you do not specify a *network_name*, it defaults to *srvname*.

Examples

1. `sp_addserver GATEWAY`

   Adds an entry for a remote server named GATEWAY in *master.dbo.sysservers*. The *network_name* is also GATEWAY.

2. `sp_addserver GATEWAY, null, VIOLET`

   Adds an entry for a remote server named GATEWAY in *master.dbo.sysservers*. The *network_name* is VIOLET. If there is already a *sysservers* entry for GATEWAY with a different *network_name*, this changes the *network_name* of server GATEWAY to VIOLET.

3. `sp_addserver PRODUCTION, local`

   Adds an entry for the local server named PRODUCTION.

Comments

- The *sysservers* table identifies the name of the local server and its options, and any remote servers that the local server can communicate with.

  To execute a remote procedure call on a remote server, it must exist in the *sysservers* table.

- If *srvname* already exists as a server name in the *sysservers* table, **sp_addserver** changes its *srvnetname* to be the name you specify with *network_name*. When it does this, it advises you which server it changed, what the old network name was, and what the new network name is.
- The installation or upgrade process for your server adds an entry in *sysservers* for a Backup Server. If you remove this entry, you cannot back up your databases.
- SQL Server requires that the Backup Server have a *srvname* of SYB_BACKUP. If you do not want to use that as the name of your Backup Server, or if you have more than one Backup Server running on your system, modify the *network_name* for server SYB_BACKUP with **sp_addserver** so SQL Server can communicate with the desired Backup Server for database dumps and loads.
- If you specify a *srvname* and a *network_name* that already exist in *sysservers*, **sp_addserver** prints an error message and does not update *sysservers*.
- Use **sp_serveroption** to set or clear server options.

Messages

- `Can’t run sp_addserver from within a transaction.`

  sp_addserver modifies the system table *master.dbo.sysservers*, so it cannot be run within a transaction.

- `Adding server '`*`srvname`*`', physical name '`*`network_name`*`'`

  The procedure was successful; *srvname* is now known to the local SQL Server and can access the physical device *network_name*.

- `Changing physical name of server '`*`srvname`*`' from '`*`old_netname`*`' to '`*`network_name`*`'`

  The server known to your SQL Server as *srvname* now accesses physical device *network_name*, instead of *old_netname*.

- `'`*`srvname`*`' is not a valid name.`

  srvname does not conform to the rules for identifiers.

- `There is already a local server.`

  Although there may be many remote servers, there can be only one local server. **sp_addserver** with the **local** option defines the name of the local server. If it already exists, the request is rejected.

- `There is already a server named '`*`srvname`*`', physical name '`*`network_name`*`'.`

  You have specified a *srvname* and *network_name* that already exist in *sysservers.* Nothing changed.

- `sp_addserver `*`servername`*` [, local | null] [, physical_name]`

  If you specify a *network_name,* you must also specify **local** or **null**.

- `Server added.`

  You have successfully added a new server.

Permissions

Only a System Security Officer can execute **sp_addserver**.

Tables Used

master.dbo.sysservers, *sysobjects*

See Also

System procedures	**sp_addremotelogin, sp_dropremotelogin, sp_dropserver, sp_helpremotelogin, sp_helpserver, sp_serveroption**

sp_addthreshold

Function

Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, SQL Server executes the associated stored procedure.

Syntax

```
sp_addthreshold database, segment, free_pages,
   procedure
```

Parameters

database – is the database for which to add the threshold. This must be the name of the current database.

segment – is the segment for which to monitor free space. Use quotes when specifying the "default" segment.

free_pages – is the number of free pages at which the threshold is crossed. When free space in the segment falls below this level, SQL Server executes the associated stored procedure.

procedure – is the stored procedure to execute when the amount of free space on *segment* drops below *free_pages*. The procedure can be located in any database on the current SQL Server or on an Open Server. Thresholds cannot execute procedures on remote SQL Servers.

Examples

1.
```
sp_addthreshold mydb, segment1, 200, pr_warning
```

   Creates a threshold for *segment1*. When free space on *segment1* drops below 200 pages, SQL Server executes the procedure **pr_warning**.

2.
```
sp_addthreshold userdb, user_data, 100,
     o_server...mail_me
```

   Creates a threshold for the *user_data* segment. When free space on *user_data* falls below 100 pages, SQL Server executes a remote procedure call to the OpenServer **mail_me** procedure.

Comments

- See the *System Administration Guide* for more information about using thresholds.

Crossing a Threshold

- When a threshold is crossed, SQL Server executes the associated stored procedure. SQL Server uses the following search path for the threshold procedure:
  - If the procedure name does not specify a database, SQL Server looks in the database in which the threshold was crossed.
  - If the procedure is not found in this database and the procedure name begins with **sp_**, SQL Server looks in the *sybsystemprocs* database.

  If the procedure is not found in either database, SQL Server sends an error message to the error log.
- SQL Server uses a **hysteresis value**, the global variable *@@thresh_hysteresis*, to determine how sensitive thresholds are to variations in free space. Once a threshold executes its procedure, it is deactivated. The threshold remains inactive until the amount of free space in the segment rises to *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space.

The Last-Chance Threshold

- By default, SQL Server monitors the free space on the segment where the log resides and executes **sp_thresholdaction** when the amount of free space is less than that required to permit a successful dump of the transaction log. This amount of free space, which is called the "last-chance threshold," is calculated by SQL Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, SQL Server suspends the transaction until log space is freed. Use **sp_dboption** to change this behavior for a particular database. Setting the **abort tran on log full** option to **true** causes SQL Server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.

Creating Additional Thresholds

- Each database can have up to 256 thresholds, including the last-chance threshold.
- When you add a threshold, it must be at least 2 times *@@thresh_hysteresis* pages from the closest threshold.

Creating Threshold Procedures

- Any user with **create procedure** permission can create a threshold procedure in a database. Usually, a System Administrator creates **sp_thresholdaction** in the *master* database, and Database Owners create threshold procedures in user databases.
- **sp_addthreshold** does not verify that the specified procedure exists. It is possible to add a threshold before creating the procedure it executes.
- SQL Server passes four parameters to a threshold procedure:
  - *@dbname*, *varchar*(30), which identifies the database
  - *@segmentname*, *varchar*(30), which identifies the segment
  - *@space_left*, *int*, which indicates the number of free pages associated with the threshold
  - *@status*, *int*, which has a value of 1 for last-chance thresholds and 0 for other thresholds

  These parameters are passed by position rather than by name; your threshold procedure can use other names for them, but must declare them in the order shown and with the correct datatypes.
- It is not necessary to create a different procedure for each threshold. To minimize maintenance, you can create a single threshold procedure in the *sybsystemprocs* database that all thresholds on the SQL Server execute.
- Include **print** and **raiserror** statements in the threshold procedure to send output to the error log.

Executing Threshold Procedures

- Tasks initiated when a threshold is crossed execute as background tasks. These tasks do not have an associated terminal or user session. If you execute **sp_who** while these tasks are running, the *status* column shows "background".

- SQL Server executes the threshold procedure with the permissions of the user who added the threshold, at the time the user executed **sp_addthreshold**, minus any permissions that have since been revoked.
- Each threshold procedure uses one user connection, for as long as it takes to execute the procedure.

Changing or Deleting Thresholds

- Use **sp_helpthreshold** for information about existing thresholds.
- Use **sp_modifythreshold** to associate a threshold with a new threshold procedure, free-space value, or segment. (You cannot change the free-space value or segment name associated with the last-chance threshold.)

  Each time a user modifies a threshold, that user becomes the threshold owner. When the threshold is crossed, SQL Server executes the threshold with the permissions of the owner at the time he or she modified the threshold, minus any permissions that have since been revoked.
- Use **sp_dropthreshold** to drop a threshold from a segment.

Disabling Free-Space Accounting

- Use the **no free space acctg** option of **sp_dboption** to disable free-space accounting on non-log segments.
- You cannot disable free-space accounting on log segments.

◆ ***WARNING!***

System procedures cannot provide accurate information about space allocation when free-space accounting is disabled.

Creating Last-Chance Thresholds for Pre-System 10.0 Databases

- Databases do not automatically acquire a last-chance threshold when upgraded to Release 10.0. Use the **lct_admin** system function to create a last-chance threshold in an existing database.
- Only databases that store their logs on a separate segment can have a last-chance threshold. Use **sp_logdevice** to move the transaction log to a separate device.

Messages

- `Adding threshold for segment '`*`segment`*`' at '`*`pageno`*`' pages.`

  The **sp_addthreshold** command succeeded.

- `Table 'systhresholds' does not exist in database '`*`database`*`'--cannot add thresholds.`

  The *systhresholds* table is missing. This table is created when the database is created (or an upgrade to Release 10.0 is performed), and must not be removed.

- `There is no segment named '`*`segment`*`'.`

  Run **sp_helpsegment** to see a list of segment names.

- `This threshold is too close to one or more existing thresholds. Thresholds must be no closer than 128 pages to each other.`

  Execute **sp_helpthreshold** to see a list of existing thresholds and sizes.

- `A threshold at `*`pageno`*` pages is logically impossible for segment '`*`segment`*`'. Choose a value between `*`value1`*` and `*`value2`*` pages.`

  A threshold must be at least 2 times *@@thresh_hysteresis* pages from the closest threshold.

- `This procedure can only affect thresholds in the current database. Say 'use `*`database_name`*`' then run this procedure again.`

  sp_addthreshold can create thresholds only in the database you are currently using. Issue the **use** command to open the database in which you want to add a threshold. Then run **sp_addthreshold** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addthreshold**.

Tables Used

master.dbo.sysusages, sysobjects, syssegments, systhresholds

See Also

Commands	**create procedure, dump transaction**
System procedures	**sp_dboption, sp_dropthreshold, sp_helpthreshold, sp_modifythreshold, sp_thresholdaction**

sp_addtype

Function

Creates a user-defined datatype.

Syntax

```
sp_addtype typename,
   phystype [(length) | (precision [, scale])]
   [, "identity" | nulltype]
```

Parameters

typename – is the name of the user-defined datatype. Type names must conform to the rules for identifiers and must be unique for each owner in each database.

phystype – is the physical or SQL Server-supplied datatype on which to base the user-defined datatype. You can specify any SQL Server datatype except *timestamp*.

The *char*, *varchar*, *nchar*, *nvarchar*, *binary*, and *varbinary* datatypes expect a *length* in parentheses. If you do not supply one, SQL Server uses the default length of one character.

The *float* datatype expects a binary *precision* in parentheses. If you do not supply one, SQL Server uses the default precision for your platform.

The *numeric* and *decimal* datatypes expect a decimal *precision* and *scale*, in parentheses and separated by a comma. If you do not supply them, SQL Server uses a default precision of 18 and scale of 0.

Enclose physical types that include punctuation, such as parentheses or commas, within single or double quotes.

identity – indicates that the user-defined datatype has the IDENTITY property. Enclose the **identity** keyword within single or double quotes. You can specify the IDENTITY property only for *numeric* datatypes with a scale of 0.

IDENTITY columns store sequential numbers, such as invoice numbers or employee numbers, that are generated automatically by SQL Server. The value of the IDENTITY column uniquely identifies each row in a table. IDENTITY columns are not updatable and do not allow nulls.

nulltype – indicates how the user-defined datatype handles null value entries. Acceptable values for this parameter are "null", "NULL", "nonull", "NONULL", "not null", and "NOT NULL". Enclose *nulltype*s that include a blank space within single or double quotes.

If you omit both the IDENTITY property and the *nulltype*, SQL Server creates the datatype using the null mode defined for the database. By default, datatypes for which no *nulltype* is specified are created NOT NULL (that is, null values are not allowed and explicit entries are required). For ANSI compatibility, use the **sp_dboption** system procedure to set the **allow nulls by default** option to **true**. This changes the database's null mode to NULL.

Examples

1. **`sp_addtype ssn, "varchar(11)"`**

   Creates a user-defined datatype called *ssn* to be used for columns that hold social security numbers. Since the *nulltype* parameter is not specified, SQL Server creates the datatype using the database's default null mode. Notice that *varchar(11)* is enclosed in quotation marks, because it contains punctuation (parentheses).

2. **`sp_addtype birthday, "datetime", null`**

   Creates a user-defined datatype called *birthday* that allows null values.

3. **`sp_addtype temp52 "numeric(5,2)"`**

   Creates a user-defined datatype called *temp52* used to store temperatures of up to five significant digits with two places to the right of the decimal point.

4. **`sp_addtype "row_id", "numeric(10,0)", "identity"`**

   Creates a user-defined datatype called *row_id* with the IDENTITY property, to be used as a unique row identifier. Columns created with this datatype store system-generated values up to 10 digits in length.

5. **`sp_addtype systype, sysname`**

   Creates a user-defined datatype with an underlying type of *sysname*. Although you cannot use the *sysname* datatype in a **create table**, **alter table**, or **create procedure** statement, you can use a user-defined datatype that is based on *sysname*.

Comments

- **sp_addtype** creates a user-defined datatype and adds it to the *systypes* system table. Once a user-defined datatype is created, you can use it in **create table** and **alter table** statements and bind defaults and rules to it.
- Build each user-defined datatype in terms of one of the SQL Server-supplied datatypes, specifying the length, or precision and scale, as appropriate. You cannot override the length, precision, or scale in a **create table** or **alter table** statement.
- A user-defined datatype name must be unique in the database, but user-defined datatypes with different names can have the same definitions.
- If *nchar* or *nvarchar* is specified as the *phystype*, then the maximum length of columns created with the new type is the length specified in **sp_addtype** multiplied by the value of *@@ncharsize* at the time the type was added.
- Each system type has a **hierarchy**, stored in the *systypes* system table. User-defined datatypes have the same datatype hierarchy as the physical types on which they are based. In a mixed mode expression, all types are converted to a common type, the type with the lowest hierarchy.

  Use the following query to list the hierarchy for each system-supplied and user-defined type in your database:

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

Types with the IDENTITY Property

- If a user-defined datatype is defined with the IDENTITY property, all columns created from it are IDENTITY columns. You can specify either **identity** or **not null**—or neither one—in the **create** or **alter table** statement. Following are three different ways to create an IDENTITY column from a user-defined datatype with the IDENTITY property:

  ```
  create table new_table (id_col IdentType)
  ```

  ```
  create table new_table (id_col IdentType identity)
  ```

  ```
  create table new_table (id_col IdentType not null)
  ```

- When you create a column with the **create table** or **alter table** statement, you can override the *nulltype* specified with the **sp_addtype** system procedure:
  - Types specified as NOT NULL can be used to create NULL or IDENTITY columns.
  - Types specified as NULL can be used to create NOT NULL columns, but not to create IDENTITY columns.

➤ ***Note***

If you try to create a null column from an IDENTITY type, the **create** or **alter table** statement fails.

Messages

- `A type with the specified name already exists.`

  Choose a different *typename*.
- `Illegal length specified—must be between 1 and 255.`

  The length of a datatype must be between 1 and 255.
- `Illegal precision specified -- must be between 1 and 38.`

  The precision of a *numeric* or *decimal* datatype must be between 1 and 38.
- `Illegal precision specified -- must be between 1 and 48.`

  The precision of a *float* or *double precision* datatype must be between 1 and 48.
- `Illegal scale specified -- must be less than precision.`

  The scale of a *numeric* or *decimal* datatype must be between 0 and the datatype's precision.
- `Physical datatype does not allow nulls.`

  You specified that you wanted to allow null values with the *bit* datatype, which doesn't allow null values.
- `Physical datatype does not exist.`

  The *phystype* you gave is not a SQL Server datatype.

- `Physical type is fixed length. You cannot specify the length.`

  The physical datatypes that take length specifications are *char*, *nchar*, *varchar*, *nvarchar*, *binary*, and *varbinary*. You cannot change the fixed lengths of other physical datatypes.

- `Type added.`

  The **sp_addtype** command succeeded. You created a user-defined datatype that can now be used in **create table** statements, or to bind rules and defaults.

- '*typename*' `is not a valid type name.`

  typename must conform to the rules for identifiers and be unique for each owner in each database.

- `User-defined datatypes based on the ‘timestamp’ datatype are not allowed.`

  The *timestamp* datatype is based on *varbinary(8)*, which you can use instead.

- `Usage: sp_addtype` *name*, ‘*datatype*’ `[, null | nonull | identity]`

  Syntax summary. The third parameter can specify either a null type (“null”, “NULL”, “nonull”, “NONULL”, “not null”, or “NOT NULL”) or the IDENTITY property.

- `User types with the identity property must be numeric with a scale of 0.`

- `You must specify a length with this physical type.`

  You used a *phystype—char*, *nchar*, *varchar*, *nvarchar*, *binary*, or *varbinary*—that requires a length. For example, “char(10)” is acceptable, but “char” is not.

Permissions

Any user can execute **sp_addtype**.

Tables Used

master.dbo.spt_values, *master.dbo.sysdatabases*, *sysobjects*, *systypes*

See Also

Commands	**create default, create rule, create table**
System procedures	**sp_bindefault, sp_bindrule, sp_dboption, sp_droptype, sp_rename, sp_unbindefault, sp_unbindrule**
Topics	Identity Columns

sp_addumpdevice

Function

Adds a dump device to SQL Server.

Syntax

```
sp_addumpdevice {"tape" | "disk"}, device_name,
   physicalname [, size]
```

Parameters

"**tape**" – for tape drives. Enclose **tape** in quotes.

"**disk**" – is for a disk or a file device. Enclose **disk** in quotes.

device_name – is the "logical" dump device name. It must be a valid identifier. Once you add a dump device to *sysdevices*, you can specify its logical name name in the **load** and **dump** commands.

physicalname – is the physical name of the device. You can specify either an absolute pathname or a relative pathname. During dumps and loads, the Backup Server resolves relative pathnames by looking in SQL Server's current working directory. Enclose names containing non-alphanumeric characters in quotation marks. For UNIX platforms, specify a non-rewinding tape device name.

size – is the capacity of the device, specified in megabytes. OpenVMS systems ignore the *size* parameter if specified. Other platforms require this parameter for tape devices but ignore it for disk devices. The *size* should be at least five database pages (each page requires 2048 bytes for most platforms, 4096 for Stratus). We recommend that you specify a capacity that is slightly below the rated capacity for your device.

Examples

1.
   ```
   sp_addumpdevice "tape", mytapedump, "/dev/nrmt8",
   40
   ```

   Adds a 40MB tape device. Dump and load commands can reference the device by its physical name, */dev/nrmt8*, or its logical name, *mytapedump*.

2. **`sp_addumpdevice "disk", mydiskdump, "/dev/rxy1d/dump.dat"`**

   Adds a disk device named *mydiskdump*. Specify an absolute or relative path name and a file name.

Comments

- **sp_addumpdevice** adds a dump device to the *master.dbo.sysdevices* table. Tape devices are assigned a *cntrltype* of 3; disk devices a *cntrltype* of 2.
- To use an operating system file as a dump device, specify a device of type **disk** and an absolute or relative path name for the *physicalname*. Omit the *size* parameter. If you specify a relative path name, dumps are made to—or loaded from—the current SQL Server working directory at the time the dump or load command is executed.
- Ownership and permission problems can interfere with the use of disk or file dump devices. **sp_addumpdevice** adds the device to the *sysdevices* table, but does not guarantee that you can create a file as a dump device or that users can dump to a particular device.
- The **with capacity** = *megabytes* clause of the **dump database** and **dump transaction** commands can override the *size* specified with **sp_addumpdevice**. On platforms that do not reliably detect the end-of-tape marker, the Backup Server issues a volume change request after the specified number of megabytes have been dumped.
- When a dump device fails, use **sp_dropdevice** to drop it from *sysdevices*. After replacing the device, use **sp_addumpdevice** to associate the logical device name with the new physical device. This avoids updating backup scripts and threshold procedures each time a dump device fails.
- To add database devices to *sysdevices*, use the **disk init** command.

Messages

- `Can’t run sp_addumpdevice from within a transaction.`

  sp_addumpdevice modifies the system table *master.dbo.sysdevices*, so it cannot be run within a transaction.
- `'`*`device_name`*`' is not a valid name.`

  The value for *device_name* must conform to the rules for identifiers.

- `device_name may not be NULL.`

  You must specify a device name.
- `Device with same logical name already exists.`

  All dump devices must have unique logical names. There is already a device with the name supplied for the *device_name* parameter.
- `'Disk' device added.`

  The disk dump device was added successfully.
- `physicalname may not be NULL.`

  You must specify a physical dump device name.
- `Please specify media capacity in megabytes (1 MB minimum).`

  You must specify a tape capacity in megabytes for tape devices. The minimum capacity is 1MB. There is no default.
- `'Tape' device added.`

  The *tape* dump device was added successfully.
- `WARNING: physical device name '`*`physicalname`*`' is not unique.`

  You attempted to create a new dump device that has the same physical name as an existing dump device.
- `WARNING: specified size parameter is not used for the disk device type.`
- `Unknown device type. Use 'disk' or 'tape'.`

  The value supplied for the first parameter isn't a known device type.

Permissions

Only a System Administrator can execute **sp_addumpdevice**.

Tables Used

master.dbo.sysdevices, sysobjects

See Also

Commands	**disk init, dump database, dump transaction, load database, load transaction**
System procedures	**sp_dropdevice, sp_helpdevice**

sp_adduser

Function

Adds a new user to the current database.

Syntax

sp_adduser *login_name* [, *name_in_db* [, *grpname*]]

Parameters

login_name – is the user’s name as found in *master.dbo.syslogins*.

name_in_db – is a new name for the user in the current database.

grpname – adds the user to an existing group in the database.

Examples

1. **sp_adduser margaret**

   Adds “margaret” to the database. Her database user name is the same as her SQL Server login name, and she belongs to the default group, “public”.

2. **sp_adduser haroldq, harold, fort_mudge**

   Adds “haroldq” to the database. When “haroldq” uses the current database, his name is “harold.” He belongs to the “fort_mudge” group, as well as the default group “public”.

Comments

- The Database Owner executes **sp_adduser** to add a user name to the *sysusers* table of the current database, enabling the user to access the current database under his or her own name.
- Specifying a *name_in_db* parameter gives the new user a name in the database different from his or her login name on SQL Server. The ability to assign a user a different name is provided as a convenience. It is not an alias as provided by **sp_addalias** as it is not mapped to the identity and privileges of another user.
- A user can be a member of only one group other than the default group, “public”. Every user is a member of the default group, “public”. Use **sp_changegroup** to change a user’s group.
- In order to access a database, a user must either be listed in *sysusers* (with **sp_adduser**) or mapped to another user in

sysalternates (with **sp_addalias**), or there must be a "guest" entry in *sysusers*.

Messages

- `A user with the same name already exists in the database.`

  The *name_in_db* is already a user in the database. Choose another name.
- `All user ids have been assigned.`

  The database has reached the maximum number of user IDs.
- `'`*`name_in_db`*`' is not a valid name.`

  The *name_in_db* specified does not follow the rules for identifiers.
- `New user added.`

  The **sp_adduser** command succeeded. The user is now known in the current database.
- `No group with the specified name exists.`

  The group name you supplied does not exist in this database. Either omit the *grpname* parameter or create the group with **sp_addgroup**.
- `No login with the specified name exists.`

  The *login_name* you gave is unknown to SQL Server. Each user must have a login on SQL Server before being added to a database.
- `User already has a login under a different name.`

  The user with the *login_name* you supplied is listed in the current database's *sysusers* table with a name different from the one supplied as the *name_in_db* parameter.
- `User already has alias access to the database.`

  The *login_name* is already known to the database by an alias. To add the user, drop the alias with **sp_dropalias** and then re-execute **sp_adduser**.

Permissions

Only the Database Owner or a System Administrator can execute **sp_adduser**.

Tables Used

master.dbo.syslogins, *master.dbo.syssrvroles*, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**grant**, **revoke**, **use**
System procedures	**sp_addalias**, **sp_addgroup**, **sp_changegroup**, **sp_dropalias**, **sp_dropgroup**, **sp_helpuser**
Topics	Identifiers

sp_auditdatabase

Function

Establishes auditing of different types of events within a database, or of references to objects within that database from another database.

Syntax

```
sp_auditdatabase [dbname [, "ok | fail | both | off"
   [, {"d u g r t o"}]]]
```

Parameters

dbname – is the name of the database for which to establish auditing.

ok | **fail** | **both** | **off** – establishes auditing of only successful attempts (**ok**), only failed attempts (**fail**), or of all attempts (**both**) to execute the events named in the third parameter. The **fail** option audits access attempts that fail because the user lacks permission to access the database. **off** turns off the specified type of auditing on the named database.

d u g r t o – are the types of database events to audit. Choose one or more, in any order. If you do not specify an event, the **ok** | **fail** | **both** | **off** argument applies to all event types (**d**, **u**, **g**, **r**, **t**, and **o**). The event types are as follows:

Event Type	Meaning
d	Audits execution of the **drop table**, **drop view**, **drop procedure**, or **drop trigger** commands within *dbname*, and execution of the **drop database** command when *dbname* is being dropped.
u	Audits execution of the **use** command on *dbname*.
g	Audits execution of the **grant** command within *dbname*.
r	Audits execution of the **revoke** command within *dbname*.
t	Audits execution of the **truncate table** command within *dbname*.
o	"Outside access"; audits execution of SQL commands from within another database that refer to objects in *dbname*.

Table 1-2: Database Auditing Options

Examples

1. **`sp_auditdatabase`**

   Displays the current auditing status for all databases on the server.

2. **`sp_auditdatabase pubs2`**

   Displays the current auditing status for the *pubs2* database.

3. **`sp_auditdatabase pubs2, "both", "ugr"`**

   Audits both successful and failed executions of the **use** command on the *pubs2* database, and of the **grant** and **revoke** commands within *pubs2*.

4.
   ```
   sp_auditdatabase pubs2, "ok", "d"
   go
   sp_auditdatabase pubs2, "fail", "u"
   go
   sp_auditdatabase pubs2, "both", "gr"
   go
   ```

   Audits successful execution of the **drop** command within the *pubs2* database and successful attempts to drop *pubs2*, attempts to use *pubs2* which failed due to a lack of permission, and both successful and failed executions of the **grant** and **revoke** commands from within *pubs2*.

5. **`sp_auditdatabase pubs2, "off", "gru"`**

   Disables auditing of the **grant** and **revoke** commands within *pubs2*, and execution of the **use** command on *pubs2*.

6. **`sp_auditdatabase pubs2, "fail"`**

   Audits failed attempts of all six event types.

Comments

- If you execute **sp_auditdatabase** more than once on a database, the options that you set accumulate with each execution. Therefore, you can enable some options for success only, some for failure only, and some for both. This requires multiple invocations of **sp_auditdatabase**, as shown in Example 4.

Messages

- `Can’t run sp_auditdatabase from within a transaction.`

  Since **sp_auditdatabase** modifies system tables, it cannot be run from within a transaction.

- `No databases currently have auditing enabled.`

  When you execute **sp_auditdatabase** with no parameters, it returns this message or the following message and the current audit settings for all databases:

  `'`*`dbname`*`' has the following auditing options enabled:`

- *`'dbname`*`' does not exist.`

  You specified an invalid database name.

- `Audit option has been changed and has taken effect immediately.`

  The **sp_auditdatabase** command succeeded. You successfully changed the audit options.

- `Invalid second argument.  Valid choices are 'ok', 'fail', 'both', or 'off'.`

  You specified an incorrect second parameter.

- `Invalid third argument.  Valid choices are 'd', 'u', 'o', 'g', 'r', or 't'.`

  You specified an incorrect third parameter.

- `Error updating the audit flags in memory. This is a system error. Contact a System Administrator.`

  Contact a System Administrator for help.

- `Error updating the audit flags in the system catalogs. This is a system error.  Contact a user with System Administrator (SA) role.`

  Contact a System Administrator for help.

Permissions

Only a System Security Officer can execute **sp_auditdatabase**.

Tables Used

sybsecurity.dbo.sysaudits

See Also

System procedures	**sp_auditoption**
Topics	Auditing

sp_auditlogin

Function

Audits a SQL Server user's attempts to access tables and views; audits the text of a user's command batches; lists users on which auditing is enabled; gives the auditing status of a user; or displays the status of table, view, or command text auditing.

Syntax

```
sp_auditlogin [login_name [, "table" | "view"
   [, "ok" | "fail" | "both" | "off"]]]

sp_auditlogin [login_name [, "cmdtext"
   [, "on" | "off"]]]
```

Parameters

login_name – is the SQL Server login name of the user for whom to establish auditing.

table | **view** – is the **table** option audits *login_name*'s attempts to access tables in any database, or returns the status of table auditing for *login_name*. **view** audits *login_name*'s attempts to access views in any database, or returns the status of view auditing for *login_name*. Enable the **table** | **view** option for successful accesses, failed accesses (where access fails because the user doesn't have the correct permissions on the object), or both.

ok | **fail** | **both** | **off** – selectively enables auditing for successful table or view accesses only (**ok**), accesses that fail due to lack of permissions on an object (**fail**), or both successful and failed accesses (**both**). **off** disables auditing of the named type — **table** or **view**.

cmdtext – preserves the text of all command batches that *login_name* submits to the server. The text is stored in the *extrainfo* column of *sybsecurity..sysaudits*.

on | **off** – **on** enables **cmdtext** auditing for *login_name*; **off** terminates it.

Examples

1. **`sp_auditlogin`**

   Returns the login names of users for whom auditing is enabled on the current server.

2. **`sp_auditlogin "joe"`**

   Displays the auditing status of user “joe”.

3. **`sp_auditlogin "joe", "table", "fail"`**
   `sp_auditlogin "joe", "view", "fail"`

   Audits Joe’s attempts to access tables and views on which he lacks permission.

4. **`sp_auditlogin "joe", "cmdtext", "on"`**

   Audits the text of commands executed by user “joe”.

5. **`sp_auditlogin "joe", "view"`**

   Displays whether view access auditing is on or off for user “joe”.

Comments

- You must issue separate **sp_auditlogin** commands to enable both table and view auditing for a single user, as shown in Example 3.
- **sp_auditlogin** establishes auditing for a specified user at the server level, not the database level. SQL Server audits the user’s attempts to access objects in any database on the server.
- You can execute **sp_auditlogin** from within any database.
- **sp_auditlogin** can display different kinds of auditing information, depending on the number of arguments supplied:
  - Used with no arguments, it displays the login names of the server users for whom auditing is currently enabled.
  - The following syntax:

    sp_auditlogin "*login_name*"

    displays the auditing status of *login_name*.
  - The following syntax:

    sp_auditlogin "*login_name*", "table"
    sp_auditlogin "*login_name*", "view"

    displays the status of table or view auditing for *login_name.*
  - The following syntax:

    sp_auditlogin "*login_name*", "cmdtext"

    displays the status of **cmdtext** auditing for *login_name.*

Messages

- `Can’t run sp_auditlogin from within a transaction.`

  sp_auditlogin updates system tables, so it cannot be run from within a transaction.
- `*login_name* does not exist.`

  You specified an invalid *login_name*.
- `*login_name* has the following auditing options enabled:`

  Lists *login_name*’s current audit settings.
- `Invalid second argument. Valid options are 'table', 'view', or 'cmdtext'.`

  You specified an incorrect parameter.
- `No logins currently have auditing enabled.`

  When you execute **sp_auditlogin** with no parameters, it returns this message if there are no logins with auditing enabled.
- `'*parameter*' is not a valid argument.`

  You specified an incorrect parameter.

Permissions

Only a System Security Officer can execute **sp_auditlogin.**

Tables Used

sybsecurity.dbo.sysaudits

See Also

System procedures	**sp_auditoption**
Topics	Auditing

sp_auditobject

Function

Audits accesses to tables and views.

Syntax

To audit existing tables and views:

```
sp_auditobject objname, dbname
   [, {"ok" | "fail" | "both" | "off"}
       [, "{d i s u}"]]
```

To audit newly created tables and views:

```
sp_auditobject {"default table"|"default view"},
   dbname [, {"ok" | "fail" | "both" | "off"}
   [, "{d i s u}"]]
```

Parameters

objname – is the name of a table or view in the current database.

dbname – is the name of the current database, if used with the *objname* parameter; if used with the **default table** | **default view** parameter, *dbname* can be the name of any database.

ok | **fail** | **both** | **off** – enables auditing for successful accesses only (**ok**), accesses that fail due to lack of permissions (**fail**), or both successful and failed accesses (**both**). **off** disables auditing of the specified type (**table** or **view**).

d i s u – is the type of access to audit. You can specify any number of types at one time and in any order. The possible types are:

Parameter	Meaning
d	delete
i	insert
s	select
u	update

Table 1-3: Types of Object Auditing

default table | **default view** – specifies that these audit settings are to be the defaults for newly created tables or views in the specified database. These default settings do not apply to any tables or views that exist when you execute **sp_auditobject**. Until you execute **sp_auditobject** "**default table** | **default view**" for a database, tables or views created within that database do not have any auditing options set.

Examples

1. **`sp_auditobject publishers, pubs2`**

   Displays the current auditing status of the *publishers* table in the *pubs2* database.

2. **`sp_auditobject publishers, pubs2, "fail"`**

   Audits failed attempts to access the *publishers* table.

3.
```
sp_auditobject titles, pubs2, "ok", "id"
go
sp_auditobject titles, pubs2, "fail", "u"
go
```

   Audits all successful executions of **insert** and **delete** and failed attempts to execute **update** on the *titles* table.

4. **`sp_auditobject "default table"`**

   Displays the default auditing values that apply to new tables in the current database.

5. **`sp_auditobject "default table", pubs2`**

   Displays the default auditing values that apply to new tables in the *pubs2* database.

6. **`sp_auditobject "default view", pubs2, "fail", "du"`**

   Establishes auditing of failed **delete** and **update** attempts for all new views in the *pubs2* database.

Comments

- You can audit use of the **select, update, delete,** and **insert** commands on tables and views.
- If you specify **default table** or **default view** without a database name, **sp_auditobject** displays the default audit settings for tables and views for the current database.
- If you specify **default table** or **default view** with only a database name, **sp_auditobject** displays the default audit settings for tables and views in the specified database.
- Establishing default auditing options for tables or views does not affect any views or tables that exist prior to setting the default.

Messages

- `An object name must be provided.`

  Provide the name of a table or view unless you are using the **default table** | **default view** parameter.

- `Audit option has been changed and has taken effect immediately.`

- `Audit option has been changed and will take effect after a reboot.`

- `Can’t run sp_auditobject from within a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.

- `Error: An invalid letter was specified. Use only 'd', 'u', 's', or 'i'.`

  You specified an incorrect parameter.

- `Only 'default table' or 'default view' is allowed.`

  You specified an incorrect parameter.

- `Only 'ok', 'fail', 'both', or 'off' can be specified.`

  Specify the **default table** or **default view** parameter.

- `You must provide 'ok', 'fail', 'both', or 'off' preceding the 'dusi' string.`

  Specify one of the **ok** | **fail** | **both** | **off** choices.

Permissions

Only a System Security Officer can execute **sp_auditobject.**

Tables Used

sybsecurity.dbo.sysaudits

See Also

System procedures	**sp_auditoption, sp_auditsproc**
Topics	Auditing

sp_auditoption

Function

Enables or disables system-wide auditing and global audit options, or reports on the status of audit options.

Syntax

```
sp_auditoption {"all" | "enable auditing" | "logouts"
   | "server boots" | "adhoc records"}
   [, {"on" | "off"}]

sp_auditoption {"logins" | "rpc connections" |
   "roles"} [, {"ok" | "fail" | "both" | "off"}]

sp_auditoption "errors" [, {"nonfatal" | "fatal"
   | "both"}]

sp_auditoption "{sa | sso | oper | navigator |
      replication} commands"
   [, {"ok" | "fail" | "both" | "off"}]
```

Parameters

The available audit options to enable, disable, or query are:

Option	Action
all	Enables or disables all options **except enable auditing** simultaneously. **enable auditing** must be set separately. For options that allow selective auditing for successful and / or failed executions, setting **all** to **on** is equivalent to setting all options to **on** or **both**, depending on the option. Syntax: `sp_auditoption "all" [, {"on"|"off"}]`
enable auditing	Enables or disables system-wide auditing. A System Security Officer must set the **enable auditing** option to **on** before any other auditing can take place. Enabling or disabling auditing automatically generates an audit record, so that you can bracket time periods when auditing was enabled. Syntax: `sp_auditoption "enable auditing" [, {"on"|"off"}]`
logouts	Enables or disables auditing of all logouts from the server, including unintentional logouts such as dropped connections. Syntax: `sp_auditoption "logouts" [, {"on"|"off"}]`
server boots	Enables or disables generation of an audit record when the server is rebooted. Syntax: `sp_auditoption "server boots" [, {"on"|"off"}]`

Table 1-4: Global Auditing Options

Option	Action
adhoc records	Allows users to send text to the audit trail with the **sp_addauditrecord** command. Syntax: `sp_auditoption "adhoc records", {"on"|"off"}`
logins	Enables or disables auditing of successful (**ok**), failed (**fail**), or all (**both**) login attempts by all users. To audit individual users, use **sp_auditlogin**. Syntax: `sp_auditoption "logins" [, {"ok"|"fail"|"both"|"off"}]`
rpc connections	When this option is **on**, it generates an audit record whenever a user from another host connects to the local server to run a procedure via a remote procedure call (RPC). Auditing can be enabled for all connection attempts (**both**), successful attempts only (**ok**), or failed attempts only (**fail**). Syntax: `sp_auditoption "rpc connections" [, {"ok"|"fail"| "both"|off"}]`
roles	Audits the use of the **set role** command to turn roles on and off. You can enable auditing of all attempts (**both**), successful attempts only (**ok**), or failed attempts only (**fail**). (See the *System Administration Guide* for more information.) Syntax: `sp_auditoption "roles" [, {"ok"|"fail"|"both"|"off"}]`
errors	Audits fatal errors (errors that break the user's connection to the server and require the client program to be restarted), nonfatal errors, or both kinds of errors. Fatal errors do not include server internal fatal software errors (such as bus errors, and segmentation faults). In case of internal errors, information is contained in the errorlog file for the server. Syntax: `sp_auditoption "errors" [, {"nonfatal"|"fatal" |"both"|"off”}]`
{**sa** | **sso** | **oper** | **navigator** | **replication**} **commands**	Audits the use of privileged commands—those requiring one of the roles for execution. You can enable auditing for successful executions only, failed attempts (where failure is due to the user lacking the proper role), or both. See “Roles” in the *SQL Server Reference Manual* for a list of the commands that require the various roles. Syntax: `sp_auditoption "{sa|sso|oper|navigator|replication} commands" [, {"ok"|"fail"|"both"|"off"}]`

Table 1-4: Global Auditing Options (continued)

on | **off** – **on** enables auditing of the option. **off** disables auditing for the option.

ok | **fail** | **both** | **off** – enables auditing for successful attempts, failed attempts, or both when the option is one of the following: **logins**, **rpc connections**, or a role. **off** disables auditing for the option.

nonfatal | **fatal** | **both** – for the **errors** option, enables auditing of nonfatal or fatal errors, or both.

Examples

1. **`sp_auditoption`**

   or

   `sp_auditoption "all"`

   Either of these commands displays the current settings of all the available global audit options.

2. **`sp_auditoption "enable auditing", "on"`**

   Enables system-wide auditing.

3. **`sp_auditoption "server boots", "on"`**

   Establishes auditing whenever the server boots.

4. **`sp_auditoption "logins", "fail"`**

   Establishes auditing of logins that fail due to lack of permission.

5. **`sp_auditoption "rpc connections"`**

   Displays the audit status of the **rpc connections** option.

6. **`sp_auditoption "errors", "fatal"`**

   Establishes auditing of fatal errors (errors that break the user's connection to the server and require the client program to be restarted).

7. **`sp_auditoption "sa commands", "both"`**

   Establishes auditing of all commands that require the System Administrator role, whether the execution was successful or not.

Comments

- **sp_auditoption** takes effect immediately when it is executed. You do not need to reboot the server.
- The System Security Officer establishes system-wide auditing with this command:

  ```
  sp_auditoption "enable auditing", "on"
  ```

  No other auditing takes place until this option is set to **on**. An audit record is automatically generated when the **enable auditing** option is set to **on** or **off**, so that the audit trail contains audit records that bracket the periods when auditing is enabled.
- Using **sp_auditoption** with no arguments displays the current settings of all of the global audit options.
- If you specify any audit option without a further parameter, **sp_auditoption** displays the current setting for that particular

option. The exception is the **all** option. When specified without a parameter, it displays the current settings for all of the global audit options.

- The initial value of all audit options is **off**.

Messages

- `Audit option has been changed and has taken effect immediately.`

  The **sp_auditoption** command succeeded. Changes made with **sp_auditoption** take effect immediately.

- `Audit option "`*`option`*`" does not exist. Valid options are:`

  The valid options appear. You specified an invalid option.

- `Audit option "`*`option`*`" is ambiguous. Ambiguous options are:`

  You did not type enough letters of the option name to uniquely identify an option.

- `Can’t run sp_auditoption from within a transaction.`

  Because this procedure updates system tables, it cannot be run from within a transaction.

- `'`*`option`*`' is an invalid audit option string in this context.`

  You specified an invalid parameter.

- `You must provide an audit option.`

  You did not specify an audit option.

Permissions

Only a System Security Officer can execute **sp_auditoption**.

Tables Used

sybsecurity.dbo.sysauditoptions

See Also

System procedures	**sp_addauditrecord, sp_auditdatabase, sp_auditlogin, sp_auditobject, sp_auditoption, sp_auditsproc**
Topics	Auditing

sp_auditsproc

Function

Audits the execution of stored procedures and triggers.

Syntax

To establish auditing for existing stored procedures and triggers:

```
sp_auditsproc [sproc_name | "all", dbname
   [, {"ok" | "fail" | "both" | "off"}]]
```

To establish auditing for future stored procedures and triggers:

```
sp_auditsproc "default", dbname
   [, {"ok" | "fail" | "both" | "off"}]
```

Parameters

sproc_name | **all** – specifies one or more stored procedures or triggers to audit.

- *sproc_name* enables auditing for only the named stored procedure or trigger. If you specify *sproc_name* with no other parameters, it returns the auditing status of that stored procedure or trigger.
- **all** enables auditing for all stored procedures within the specified database. If you use **all** with no other parameters, it displays the auditing status of all stored procedures in the current database.

dbname – if used with the *sproc_name* | **all** parameter, *dbname* is the name of the current database. If used with the **default** parameter, *dbname* is the name of the database to audit.

ok | **fail** | **both** | **off** – selectively enables auditing for successful executions only (**ok**), executions that fail due to lack of permission (**fail**), or both success and failure (**both**). **off** disables auditing for the named procedure or trigger. **fail** applies only to stored procedures: triggers are not subject to permissions checks, so failure does not apply to them. (Use **sp_auditobject** to audit the **select, insert, update,** and **delete** commands.)

default – sets the audit state for stored procedures and triggers created after setting the default. The default does not affect procedures and triggers already in existence. If you use **default** with *dbname* but without the final parameter, it returns the default audit status for stored procedures and triggers in the named database.

Examples

1. **`sp_auditsproc`**

   Returns the names of all stored procedures and triggers being audited in the current database.

2. **`sp_auditsproc sp_dboption`**

   Returns the current auditing status of the system procedure **sp_dboption**.

3. **`sp_auditsproc sp_dboption, master, "fail"`**

   Audits failed attempts to execute **sp_dboption** in the *master* database.

4. **`sp_auditsproc "all", pubs2`**

   Returns the auditing status of all stored procedures and triggers in the *pubs2* database.

5. **`sp_auditsproc "all", pubs2, "fail"`**

   Audits all executions of stored procedures and triggers on the current database that fail due to lack of permission.

6. **`sp_auditsproc "default"`**

   Returns the default settings for newly created stored procedures and triggers in the current database.

7. **`sp_auditsproc "default", pubs2, ok`**

   Sets a default in the *pubs2* database so that successful executions of new stored procedures and triggers are audited.

Comments

- If you execute **sp_auditsproc** with no parameters, it returns the names of any stored procedures and triggers on which auditing is currently enabled within the current database.
- **sp_auditsproc** audits the execution of stored procedures and triggers. Any parameter values passed to a procedure are also audited.

Messages

- `A sproc/trigger name or 'all' must be provided.`
- `Can’t run sp_auditsproc from within a transaction.`

  This procedure modifies system tables, so it cannot be run from within a transaction.
- `No databases currently have default sproc/trigger auditing enabled.`
- `No sprocs/triggers currently have auditing enabled.`
- `Only 'ok', 'fail', 'both' or 'off' can be specified.`

  You specified an invalid argument.
- *sproc_name* `does not exist.`

  You specified an invalid stored procedure name.
- `'`*sproc_name*`' has the following auditing options enabled:`

  sp_auditsproc *sproc_name* returns a list of the audit options on the specified stored procedure or trigger.
- `The third argument was not necessary; therefore, it was ignored.`

Permissions

Only a System Security Officer can execute **sp_auditsproc.**

Tables Used

sybsecurity.dbo.sysaudits

See Also

System procedures	**sp_auditoption**
Topics	Auditing

sp_bindefault

Function

Binds a default to a column or user-defined datatype.

Syntax

sp_bindefault *defaultname*, *objectname* [, futureonly]

Parameters

defaultname – is the name of a default created with **create default** statements to bind to specific columns or user-defined datatypes.

objectname – is the name of the table and column, or user-defined datatype, to which to bind the default. If the *objectname* parameter is not of the form *"table.column"*, it is assumed to be a user-defined datatype. If the object name includes embedded blanks or punctuation, or is a reserved word, enclose it in quotation marks.

By default, existing columns of the user-defined datatype inherit the default *defaultname,* unless the column's default was previously changed.

futureonly – prevents existing columns of a user-defined datatype from acquiring the new default. This parameter is optional when binding a default to a user-defined datatype. It is never used when binding a default to a column.

Examples

1. **sp_bindefault today, "employees.startdate"**

   Assuming that a default named *today* has been defined in the current database with **create default**, this command binds it to the *startdate* column of the *employees* table. Each new row added to the *employees* table has the value of the *today* default in the *startdate* column unless another value is supplied.

2. **sp_bindefault def_ssn, ssn**

   Assuming that a default named *def_ssn* and a user-defined datatype named *ssn* exist, this command binds *def_ssn* to *ssn*. The default is inherited by all columns that are assigned the user-defined datatype *ssn* when a table is created. Existing columns of type *ssn* also inherit the default *def_ssn* unless you specify **futureonly** (which prevents existing columns of that user-defined datatype from inheriting the default), or unless the column's default has previously been changed (in which case the changed default is maintained).

3. **`sp_bindefault def_ssn, ssn, futureonly`**

   Binds the default *def_ssn* to the user-defined datatype *ssn*. Because the **futureonly** parameter is included, no existing columns of type *ssn* are affected.

Comments

- You can create column defaults in two ways: by declaring the default as a column constraint in the **create table** or **alter table** statement, or by creating the default using the **create default** statement and binding it to a column using **sp_bindefault**. Using **create default**, you can bind that default to more than one column in the database.
- You cannot bind a default to a SQL Server-supplied datatype.
- Defaults bound to a column or user-defined datatype with the IDENTITY property have no effect on column values. Each time you insert a row into the table, SQL Server assigns the next sequential number to the IDENTITY column.
- If binding a default to a column, give the *objectname* argument in the form *"table.column"*. Any other format is assumed to be the name of a user-defined datatype.
- If a default already exists on a column, you must remove it before binding a new default. Use **sp_unbindefault** to remove defaults created with **sp_bindefault**. Use **alter table** to remove defaults created with **create table**.
- Existing columns of the user-defined datatype inherit the new default unless their default was previously changed, or the value of the optional third parameter is **futureonly**. New columns of the user-defined datatype always inherit the default.
- Statements that use a default cannot be in the same batch as their **sp_bindefault** statement.

Messages

- `Default and table or usertype must be in current database.`

  The *objectname* parameter supplied with the procedure contained a reference to another database. Defaults can be bound to objects in the current database only.

- `Default bound to column.`

  The default was successfully bound to the specified column in the specified table.

- `Default bound to datatype.`

  The default was successfully bound to the specified user-defined datatype.

- `No such default exists. You must create the default first.`

  First create the default in the current database with **create default**. Then execute **sp_bindefault**.

- `The column already has a default. Bind disallowed.`

  Execute **sp_unbindefault** to unbind the existing default.

- `The new default has been bound to column(s) of the specified user datatype.`

  The command succeeded. Existing columns of the user-defined datatype specified now have the new default bound to them (unless their defaults were previously changed).

- `Usage: sp_bindefault defaultname, objectname [, 'futureonly']`

  Syntax summary. You incorrectly specified a parameter to **sp_bindefault.**

- `You cannot bind a declared default. The default must be created using create default.`

  First create the default in the current database with **create default**. Then execute **sp_bindefault.**

- `You can’t bind a default to a timestamp datatype column.`

  The value in a *timestamp* column represents a SQL Server-supplied sequence identifier. You cannot supply a default value for a timestamp.

- `You do not own a column of that name.`

  Only the owner of a table can bind a default to any of its columns. You are not the owner, or the object doesn’t exist.

- `You do not own a datatype of that name.`

  Only the owner of a user-defined datatype can bind a default to it. You are not the owner.

Permissions

Only the object owner can issue **sp_bindefault**.

Tables Used

syscolumns, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create default**, **create table**, **drop default**
System procedures	**sp_unbindefault**

sp_bindmsg

Function

Binds a user message to a referential integrity constraint or check constraint.

Syntax

```
sp_bindmsg constraint_name, message_num
```

Parameters

constraint_name – is the name of the integrity constraint to which you are binding a message. Use the **constraint** clause of the **create table** command, or the **add constraint** clause of the **alter table** command to create and name constraints.

message_num – is the number of the user message to bind to an integrity constraint. The message must exist in the *sysusermessages* table in the local database prior to calling **sp_bindmsg**.

Examples

1. `sp_bindmsg positive_balance, 20100`

   Binds user message number 20100 to the *positive_balance* constraint.

Comments

- **sp_bindmsg** binds a user message to an integrity constraint by adding the message number to the constraint row in the *sysconstraints* table.
- Only one message can be bound to a constraint. To change the message for a constraint, just bind a new message. The new message number replaces the old message number in the *sysconstraints* table.
- Use the **sp_addmessage** procedure to insert user messages into the *sysusermessages* table.
- **sp_help** *tablename* displays all constraint names declared on *tablename*.
- **sp_getmessage** procedure retrieves message text from the *sysusermessages* table.

Messages

- `Binding message failed unexpectedly. Please try again.`

  An error occurred while binding this message. Reissue the command.

- `Constraint name must be in 'current' database.`

  You can only bind messages to constraints that are defined in the current database.

- `Constraint name must belong to the current user.`

  You cannot bind a message to a constraint created by another user.

- `Message bound to constraint`

  You successfully bound the message to the constraint.

- `Message id must be a user defined message.`

  User-defined messages must have a number greater than 20000. Only user-defined messages can be bound to constraints.

- `No such constraint exists. Please create the constraint first using CREATE/ALTER TABLE command.`

  Use **create table** or **alter table** to create the constraint before binding a message to it. You can see a list of all existing constraints on a table by using **sp_help** *tablename*.

- `No such message exists. Please create the message first using sp_addmessage.`

  The message must exist in the *sysusermessages* table before you can bind it to a constraint. Use **sp_addmessage** to create the message.

- `No such referential or check constraint exists. Please check whether the constraint name is correct.`

  You can see a list of all existing constraints on a table by using **sp_help** *tablename*.

Permissions

Only the object owner can execute **sp_bindmsg.**

Tables Used

sysconstraints, sysobjects, sysusermessages

See Also

Commands	**alter table, create table**
System procedures	**sp_addmessage, sp_getmessage, sp_unbindmsg**

sp_bindrule

Function

Binds a rule to a column or user-defined datatype.

Syntax

```
sp_bindrule rulename, objectname [, futureonly]
```

Parameters

rulename – is the name of a rule. Create rules with **create rule** statements and bind rules to specific columns or user-defined datatypes with **sp_bindrule**.

objectname – is the name of the table and column, or user-defined datatype, to which the rule is to be bound. If *objectname* is not of the form *"table.column"*, it is assumed to be a user-defined datatype. If the object name has embedded blanks or punctuation, or is a reserved word, enclose it in quotation marks.

futureonly – prevents existing columns of a user-defined datatype from inheriting the new rule. This parameter is optional when binding a rule to a user-defined datatype. It is meaningless when binding a rule to a column.

Examples

1. **`sp_bindrule today, "employees.startdate"`**

   Assuming that a rule named *today* has been created in the current database with **create rule**, this command binds it to the *startdate* column of the *employees* table. When a row is added to *employees*, the data for the *startdate* column is checked against the rule *today*.

2. **`sp_bindrule rule_ssn, ssn`**

   Assuming the existence of a rule named *rule_ssn* and a user-defined datatype named *ssn*, this command binds *rule_ssn* to *ssn*. In a **create table** statement, columns of type *ssn* inherit the rule *rule_ssn*. Existing columns of type *ssn* also inherit the rule *rule_ssn*, unless *ssn*'s rule was previously changed (in which case the changed rule is maintained in the future only).

3. **`sp_bindrule rule_ssn, ssn, futureonly`**

   The rule *rule_ssn* is bound to the user-defined datatype *ssn*, but no existing columns of type *ssn* are affected. **futureonly** prevents existing columns of type *ssn* from inheriting the rule.

Comments

- First use the **create rule** statement to create a rule. Then execute **sp_bindrule** to bind it to a column or user-defined datatype in the current database.

  The rule is enforced when an **insert** is attempted, not at binding. You can bind a character rule to a column with an exact or approximate numeric datatype, even though such an **insert** is illegal.
- You cannot use **sp_bindrule** to bind a check constraint for a column in a **create table** statement.
- You cannot bind a rule to a SQL Server-supplied datatype, or to a *text* or *image* column.
- If binding to a column, the *objectname* argument must be of the form "*table.column*". Any other format is assumed to be the name of a user-defined datatype.
- Statements that use a rule cannot be in the same batch as their **sp_bindrule** statement.
- You can bind a rule to a column or user-defined datatype without unbinding an existing rule. Rules bound to columns always take precedence over rules bound to user-defined datatypes. Binding a rule to a column will replace a rule bound to the user-defined datatype of that column, but binding a rule to a datatype will not replace a rule bound to a column of that user-defined datatype. The following chart indicates the precedence when binding rules to columns and user-defined datatypes where rules already exist:

	Old Rule Bound to	
New Rule Bound to	user-defined datatype	column
user-defined datatype	replaces old rule	no change
column	replaces old rule	replaces old rule

Table 1-5: Precedence of New and Old Bound Rules

- Existing columns of the user-defined datatype inherit the new rule unless their rule was previously changed, or the value of the optional third parameter is **futureonly.** New columns of the user-defined datatype always inherit the rule.

Messages

- `No such rule exists. You must create the rule first.`

  First create the rule in the current database with **create rule**. Then execute **sp_bindrule**.

- `Rule and table or usertype must be in current database.`

  The *objectname* parameter contained a reference to another database. Rules can only be bound to objects in the current database.

- `Rule bound to datatype.`

  The rule was successfully bound to the specified user-defined datatype.

- `Rule bound to table column.`

  The rule was successfully bound to the specified column in the specified table.

- `The new rule has been bound to column(s) of the specified user datatype.`

  Existing columns of the specified user-defined datatype now have the new rule bound to them (unless their rules were previously changed).

- `Usage: sp_bindrule rulename, objectname [,futureonly]`

  Syntax summary. You incorrectly specified a parameter to **sp_bindrule.**

- `You can’t bind a rule to a text, image, or timestamp datatype column.`

  The column you specified was a *text*, *image*, or *timestamp* column. Rules cannot be applied to *text*, *image*, or *timestamp* datatypes.

- `You can’t bind a rule to a text, image, or timestamp datatype.`

  The datatype you specified was a *text, image,* or *timestamp* datatype. Rules cannot be applied to *text, image,* or *timestamp* datatypes.

- `You cannot bind a declared constraint. The rule must be created using create rule.`

  First create the rule in the current database with **create rule**. Then execute **sp_bindrule**.

- `You do not own a column of that name.`

  Only the owner of a table can bind a rule to any of its columns. You are not the owner, or the object doesn't exist.

- `You do not own a datatype of that name.`

  Only the owner of a user-defined datatype can bind a rule to it. You are not the owner.

Permissions

Only the object owner can execute **sp_bindrule**.

Tables Used

syscolumns, *sysconstraints*, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create rule, drop rule**
System procedures	**sp_unbindrule**

sp_changedbowner

Function

Changes the owner of a database. **Do not** change the owner of the *sybsystemprocs* database.

Syntax

```
sp_changedbowner login_name [, true ]
```

Parameters

login_name – is the login name of the new owner of the current database. The new owner must not already be known as either a user or alias (that is, the new owner must not already be listed in *sysusers* or *sysalternates*). Executing **sp_changedbowner** with the single parameter *login_name* changes the database ownership to *login_name* and drops aliases of users who could act as the old “dbo.”

true – To transfer aliases and their permissions to the new “dbo,” add this optional parameter. The only acceptable values are “true” or “TRUE”.

Examples

1. `sp_changedbowner albert`

   Makes the user “albert” the owner of the current database.

Comments

- After executing **sp_changedbowner**, the new owner is known as Database Owner inside the database.
- The new owner must already have a login name on SQL Server, but must **not** have a database user name or alias name in the database. To assign Database Ownership to such a user, drop the user name or alias entry before executing **sp_changedbowner**.
- To grant permissions to the new owner, a System Administrator must grant them to the Database Owner, since the user is no longer known inside the database under any other name.

Messages

- `Can’t change the owner of the master database.`

  No one can change the owner of the *master* database.

- `Database owner changed.`

  The **sp_changedbowner** command succeeded and the Database Owner changed.

- `Only the System Administrator (SA) or the Database Owner (dbo) can change the owner of a database.`

  You must be a System Administrator or the Database Owner to execute **sp_changedbowner**.

- `The dependent aliases were mapped to the new dbo.`

  You set the optional parameter "true". Aliases and their permissions transferred to the new "dbo".

- `The dependent aliases were dropped.`

  You did not set the optional parameter "true". Aliases and their permissions have been dropped.

- `No login with the specified name exists.`

  The proposed new Database Owner must have a login on SQL Server.

- `The proposed new db owner already is a user in the database.`

  The specified *login_name* is already a user in the current database. To make the user the Database Owner, drop the user entry from the current database's *sysusers* table.

- `The proposed new db owner already is aliased in the database.`

  The specified *login_name* is already aliased in the current database. To make the user the Database Owner, drop the user alias entry from the current database's *sysalternates* table.

Permissions

Only a System Administrator or the Database Owner can execute **sp_changedbowner.**

Tables Used

master.dbo.syslogins, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**create database**
System procedures	**sp_addlogin, sp_dropalias, sp_dropuser, sp_helpdb**

sp_changegroup

Function

Changes a user's group.

Syntax

```
sp_changegroup grpname, name_in_db
```

Parameters

grpname – is the name of the group. The group must already exist in the current database. If you use "public" as the *grpname*, enclose it in quotes because it is a SQL keyword.

name_in_db – is the name of the user to add to the group. The user must already exist in the current database.

Examples

1. `sp_changegroup fort_mudge, albert`

   The user "albert" is now a member of the "fort_mudge" group. It doesn't matter what group "albert" belonged to before.

2. `sp_changegroup "public", albert`

   Removes "albert" from the group he belonged to without making him a member of a new group (all users are always members of "public".)

Comments

- Executing **sp_changegroup** adds the specified user to the specified group. The user is dropped from the group he or she previously belonged to and added to the one specified by *grpname*.
- New database users can be added to groups at the same time they are given access to the database with **sp_adduser**.
- Groups are used as a collective name for granting and revoking privileges. Every user is always a member of the default group, "public", and can belong to only one other group.
- To remove someone from a group without making him a member of a new group, use this command:

  ```
  sp_changegroup "public", name_in_db
  ```

- When a user changes from one group to another, the user loses all permissions that he or she had as a result of belonging to the old group, and gains the permissions granted to the new group.

Messages

- `Group changed.`

  The user now belongs to the specified group.

- `No group with the specified name exists.`

  The specified group doesn't exist in the current database.

- `No user with the specified name exists in the current database.`

  The specified user doesn't exist in the current database.

Permissions

Only the Database Owner or a System Administrator can execute **sp_changegroup**.

Tables Used

master.dbo.syssrvroles, *syscolumns*, *sysobjects*, *sysprotects*, *sysusers*

See Also

Commands	**grant, revoke**
System procedures	**sp_addgroup, sp_adduser, sp_dropgroup, sp_helpgroup**

sp_checknames

Function

Checks the current database for names that contain characters not in the 7-bit ASCII set.

Syntax

`sp_checknames`

Parameters

None.

Examples

1. **`sp_checknames`**

```
Looking for non 7-bit ASCII characters in the system tables
of database:
“master”


===============================================================
Table.Column name:  “syslogins.password”

The following logins have passwords that contain non 7-bit
ASCII characters.  If you wish to change them use “sp_password”;
Remember, only the sa and the login itself may examine or change
the syslogins.password column:

 suid   name
 ------ ------------------------------
      1 sa
      2 probe
      3 bogususer
```

Comments

- **sp_checknames** examines the names of all objects, columns, indexes, user names, group names, and other elements in the current database for characters outside of the 7-bit ASCII set. It reports illegal names and gives instructions to make them compatible with the 7-bit ASCII set.
- Run **sp_checknames** in every database on your server after upgrading from a server of release 4.0.x or 4.2.x, and using a default character set that was not 7-bit ASCII.

- Follow the instructions in the **sp_checknames** report to correct all of the non-ASCII names.

Messages

- `Good news?  Database "master" has no obj/user/etc. names that contain non 7-bit ASCII characters.`

  If **sp_checknames** finds any names are found that are not fully 7-bit ASCII, appropriate messages and remedial instructions appear.

Permissions

Any user can execute **sp_checknames**.

Tables Used

sp_checknames uses the following tables when run in any database:

dbo.syscolumns, *dbo.sysindexes*, *dbo.sysobjects*, *dbo.syssegments*, *dbo.systypes*, *dbo.sysusers*

sp_checknames uses the following tables only when run in the *master* database:

master.dbo.sysdatabases, master.dbo.sysdevices, master.dbo.syslogins, master.dbo.sysremotelogins, master.dbo.sysservers

See Also

Commands	**update**
System procedures	**sp_defaultdb, sp_password, sp_rename, sp_renamedb**

sp_checkreswords

Function

Detects and displays identifiers that are Transact-SQL reserved words. Checks server names, device names, database names, segment names, user-defined datatypes, object names, column names, user names, login names, and remote login names.

Syntax

sp_checkreswords [*username*]

Parameters

username – is the name of a user in the current database. If you supply *username*, **sp_checkreswords** checks only for objects that the specified user owns.

Examples

1. **sp_checkreswords** *(executed in master)*

```
Reserved Words Used as Database Object Names for Database master

Upgrade renames sysobjects.schema to sysobjects.schemacnt.

Owner
------------------------------
dbo

Table                          Reserved Word Column Names
------------------------------ ------------------------------
authorization                  cascade

Object Type                    Reserved Word Object Names
------------------------------ ------------------------------
rule                           constraint
stored procedure               check
user table                     arith_overflow
user table                     authorization

-------------------------------------------------------------
-------------------------------------------------------------

Owner
------------------------------
lemur
```

```
Table                          Reserved Word Column Names
------------------------------ ------------------------------
key                            close

Table                          Reserved Word Index Names
------------------------------ ------------------------------
key                            isolation

Object Type                    Reserved Word Object Names
------------------------------ ------------------------------
default                        isolation
rule                           level
stored procedure               mirror
user table                     key

Reserved Word Datatype Names
------------------------------
identity


-------------------------------------------------------------
-------------------------------------------------------------

Database-wide Objects
---------------------


Reserved Word User Names
------------------------------
at
identity

Reserved Word Login Names
------------------------------
at
identity

Reserved Word as Database Names
------------------------------
work

Reserved Word as Language Names
------------------------------
national

Reserved Word as Server Names
```

```
 ------------------------------
 mirror
 primary

 Reserved Word ServerNetNames
 --------------------------------
 mirror
 primary

(return status = 22)
```

2. sp_checkreswords *(executed in user database***)**

```
Reserved Words Used as Database Object Names for Database user_db

 Upgrade renames sysobjects schema to sysobjects.schemacnt.

 Owner
 ------------------------------
 tamarin

 Table                          Reserved Word Column Names
 ------------------------------ ------------------------------
 cursor                         current
 endtran                        current
 key                            identity
 key                            varying
 schema                         primary
 schema                         references
 schema                         role
 schema                         some
 schema                         user
 schema                         work

 Table                          Reserved Word Index Names
 ------------------------------ ------------------------------
 key                            double

 Object Type                    Reserved Word Object Names
 ------------------------------ ------------------------------
 default                        escape
 rule                           fetch
 stored procedure               foreign
 user table                     cursor
 user table                     key
 user table                     schema
 view                           endtran
 ------------------------------------------------------------
 ------------------------------------------------------------

 Database-wide Objects
 --------------------

Found no reserved words used as names for database-wide objects.

(return status = 18)
```

Comments

- Use **sp_checkreswords** before or immediately after upgrading to a new version of SQL Server. See the *Sybase SQL Server Installation Guide* for your platform for information on installing and running this procedure before performing the upgrade.
- **sp_checkreswords** also finds reserved words used as identifiers that were created using the **set quoted_identifier** option.
- Run **sp_checkreswords** in *master* and each of your user databases. Also run it in *model* if you have added users or objects to the *model* database.
- The return status indicates the number of items found.
- **sp_checkreswords** reports the names of existing objects that are reserved words. Transact-SQL does not allow words that are part of any command syntax to be used for identifiers, unless you are using delimited identifiers. Reserved words are pieces of SQL syntax, and they have special meaning when you type them as part of a command. For example, in a pre-System 10.0 server, you could have a table called *work*, and select data from it with this query:

  ```
  select * from work
  ```

  work is a new reserved word in System 10.0, part of the command **commit work.** Typing the same **select** statement in a System 10.0 SQL Server causes a syntax error. **sp_checkreswords** finds identifiers that would cause these problems.
- If you supply a user name, **sp_checkreswords** checks for all of the objects that a user can own: tables, indexes, views, procedures, triggers, rules, defaults, and user-defined datatypes. It reports all identifiers that are reserved words.
- If your current database is a user database, *model*, or *tempdb*, and you do not provide a user name, **sp_checkreswords** checks for all of

the objects above, with a separate section in the report for each user name. It also checks *sysusers* and *syssegments* for user names and segment names that are reserved words. You only need to check *model* if you have added objects, users, or user-defined datatypes to *model*.

- If your current database is *master*, and you do not provide a user name, **sp_checkreswords** performs all of the checks above and also checks *sysdatabases*, *syslogins*, *syscharsets*, *sysservers*, *sysremotelogins*, *sysdevices* and *syslanguages* for reserved words used as the names of databases, local or remote logins, local and remote servers, character sets or languages.

Handling Reported Instances of Reserved Words

- If **sp_checkreswords** reports that reserved words are used as identifiers, you have two options:
  - Change the name of the identifier using **sp_rename, sp_renamedb**, or, in some cases, by performing updates to system tables.
  - Use the **quoted_identifier** option of the **set** command if the reserved word is a table name, view name, or column name. If most of your applications use stored procedures, you can drop and re-create these procedures with the **quoted_identifier** option set, and all identifiers quoted. All users will be able to run them, without having to turn the **quoted_identifier** option on for their session. You can also turn on the **quoted_identifier** option, create views that give alternative names to tables or columns, and change your applications to reference the view instead. The following example provides alternatives for the new reserved words "key", "level", and "work":

    ```
    create view keyview
    as
    select lvl = "level", wrk = "work"
    from "key"
    ```
- If you do not change the identifiers, or use delimited identifiers, any query that uses the reserved words as identifiers reports an error, usually a syntax error. For example:

  ```
  select level, work from key
  ```

  ```
  Msg 156, Level 15, State 1:
  Server 'rosie', Line 1:
  Incorrect syntax near the keyword 'level'.
  ```

You can ignore reserved words used as identifiers only if no queries of any kind ever reference the identifier. This is probably impossible to avoid.

➤ ***Note***

The quoted identifier option is a SQL 92 option, and may not be supported by many client products which support other SQL Server features. For example, you cannot use **bcp** on tables whose names are reserved words.

Before choosing the quoted identifier option, perform a test on various objects using all of the tools you will use to access SQL Server. Turn on the quoted identifier option, and create a table with a reserved word for a name, and reserved-word column names. If the client product generates SQL code, it must enclose identifiers in double quotes (if they are reserved words) and character constants in single quotes.

- Procedures, triggers and views that depend on objects whose names have changed may continue to work for some time after the name change, and then suddenly stop working when the query plan is recompiled. Recompilation takes place for many reasons, without notification to the user. Change the names of objects in procedures, triggers, or views immediately after you change the object name.
- Whether you choose to change the object names or use delimited identifiers, you must change all stored procedures, views, triggers, and applications that include the reserved word. If you change object names, you must change identifiers; if you use delimited dentifiers, you must add the **set quoted_identifier** option and quotation marks.
- If you do not have the text of your procedures, triggers, views, rules or defaults saved in operating system files, you can use **defncopy** to copy the definitions from the server to files. See **defncopy** in the *SQL Server Utility Programs* manual for your platform.

Changing Identifiers

- If you choose to change the names of the items reported by **sp_checkreswords**, you must change the names in all of the procedures, triggers, views and applications that reference the object using the reserved word.

- Dump your database before changing identifier names. After you change the identifier names, run **dbcc** to determine that there are no problems, and dump the database again.
- If you are changing identifiers on an active production database:
  - Perform these changes when the system is least busy, so that you will disrupt as few users as possible.
  - Prepare carefully by finding all Open Client DB-Library programs, windowing applications, stored procedures, triggers, and scripts that use a particular identifier. This way, you can make the edits needed in the source code, and then change the identifiers and replace the procedures and code as quickly as possible.
- The procedure **sp_depends** can help find procedures, views, and triggers that use table and view names.

Using sp_rename to Change Identifiers

- The system procedure **sp_rename** renames tables, indexes, views, procedures, triggers, rule, defaults, user-defined datatypes, and columns. Use **sp_renamedb**, explained later, for renaming databases.
- Table 1-6 shows the types of identifiers that you can change with **sp_rename**. See the **sp_rename** examples immediately after the table. The table lists other changes that may have to be made on the server and in your application programs.

Identifier	Considerations
table name	Drop all procedures, triggers and views that reference the table, and re-create them with the new name. Use **sp_depends** to find the objects that depend on the table. Change all applications or SQL source scripts that reference the table to use the new table name. Change **dbcc** scripts that perform table-level checks using table names.

Table 1-6: sp_rename and Changing Identifiers

Identifier	Considerations
index name	Drop any stored procedures that create or drop the index, and re-create them with the new name. Change all applications or SQL source scripts that create or drop the index. Change **dbcc** scripts that perform index-level checks using index names.
view name	Drop all procedures, triggers, and views that reference the view, and re-create them with the new name. Use **sp_depends** to find the objects that depend on the view. Change all applications or SQL source scripts that reference the view to use the new view name.
procedure name	Drop and re-create with the new procedure name all procedures and triggers that reference the procedure. Change all applications or SQL source scripts that execute the procedure to use the new name. If another server remotely calls the procedure, change applications on the remote server to use the new procedure name.
trigger name	Change any SQL source scripts that create the trigger.
rule name	Change any SQL source scripts that create the rule.
default name	Change any SQL source scripts that create the default.
user-defined datatype name	Drop all procedures that create tables with user-defined datatypes, and re-create them with the new name. Change any applications that create tables with user-defined datatypes.

column name	Drop all procedures, triggers and views that reference the column, and re-createthem with the new column name.
	
	sp_depends cannot find column name references. The following query displays the names of procedures, triggers and views that reference a column named “key”:
	`select distinct sysobjects.name`
	`from sysobjects, syscomments`
	`where sysobjects.id = syscomments.id`
	`and syscomments.text like "%key%"`
	
	Change all applications and SQL source scripts that reference the column by name.

Table 1-6: sp_rename and Changing Identifiers (continued)

The following command changes the name of the view *isolation*:

```
sp_rename "isolation", isolated
```

The following command changes the name of a column in the just-renamed *isolated* table:

```
sp_rename "isolated.key", keyname
```

- Use **sp_depends** to get a list of all of the views, procedures or triggers that reference a view, procedure or table that needs to be renamed. To use **sp_depends** after renaming an object, give the new name. For example:

```
sp_depends new_name
```

Renaming Databases with sp_renamedb

- To change the name of a database, use **sp_renamedb**. The database must be in single-user mode. Drop and recreate any procedures, triggers and views that reference the database name explicitly. See **sp_renamedb** for more information.
- If you change the database name, drop, change to reflect the new name, and re-create all stored procedures, triggers and views that include the database name to refer to objects in the database (*dbname*.[*owner*].*object_name*). Also change all applications and SQL source scripts that reference the database, either in a **use** command, or as part of a fully qualified identifier (in the form *dbname*.[*owner*].*objectname*).

  If you use scripts to run your **dbcc** commands or **dump database** and **dump transaction** commands on your databases, be sure to update those scripts.

  The following example renames the database *work*:

```
sp_dboption work, single, true
use work
checkpoint
sp_renamedb work, workdb
use master
sp_dboption workdb, single, false
use workdb
checkpoint
```

Changing Other Identifiers

- To change user names, login names, device names, remote server names, remote server user names, segment names, and character set and language names, first determine if you can drop the object or user and re-add or re-create it. If not, use **sp_configure "allow updates", 1** and **reconfigure with override** to allow updates to system catalogs. Only a System Administrator can use **sp_configure** and **reconfigure with override**.

Since errors during direct updates to system tables can create severe problems in SQL Server, check *Table 1-7: Alternatives to Direct System Tables Updates When Changing Identifiers* to determine whether you can drop the objects or users, and re-create them. *Table 1-9: Considerations When Changing Identifiers* shows possible dependencies on this set of identifiers. Check this table for possible dependencies whether you choose to upgrade by dropping and recreating objects, by using delimited identifiers, or by performing direct updates to system tables.

Identifier Type	Suggested Actions to Avoid Updates to System Tables
user names and login names	To change the name of a user with no objects, first use **sp_helprotect** *username* in each database to record the user's permissions. Then, drop the user from all of the databases (**sp_dropuser**), and drop the login (**sp_droplogin**). Then, add the new login name (**sp_addlogin**), add the new user name to the databases (**sp_adduser**), and restore the user's permissions with **grant**.
device names	If this device is completely allocated, so you will not need to use its name in a **create database** command, you can leave the name unchanged.
remote server names	Unless there are large numbers of remote login names from the remote server, drop the remote server (**sp_dropserver**) and add it with a new name (**sp_addserver**).
remote server logins	Drop the remote login with **sp_dropremotelogin,** add it with a new name using **sp_addremotelogin**, and restore the user's permission to execute procedures with **grant.**
segment names	These are rarely used, once objects have been created on the segments.
character set and language names	Languages and character sets only have reserved words as identifiers if a System Administrator has created alternative languages with **sp_addlanguage.** Drop the language with **sp_droplanguage**, and add it with a new name.

Table 1-7: Alternatives to Direct System Tables Updates When Changing Identifiers

◆ ***WARNING!***

Direct updates to system tables can be very dangerous. You can make mistakes that make it impossible for SQL Server to run, or make it impossible to access objects in your databases. Undertake this effort when you are calm and collected, and when no production activity (or very little) is taking place on the server. Use the alternative methods described above, if possible.

- The following example shows a "safe" procedure for updating a user name, with all data modification proceeded by a **begin transaction** command:

```
sp_configure "allow updates", 1

reconfigure with override

begin tran
update sysusers
set name = "workerbee"
where name = "work"
```

At this point, run the query, and check to be sure that the command affected only the row that you intended to change. The only identifier change that affects more than one row is changing the *language* name in *syslogins*.

- If the query affected only the correct row, use **commit transaction**.
- If the query affected more than one row, or the incorrect row, use **rollback transaction**, determine the source of the problem, and execute the command correctly.

◆ ***WARNING!***

Only update system tables in a single database in each user defined transaction. Do not issue a begin transaction command and then update tables in several databases. Such actions can make recovery extremely difficult.

The following table shows the system tables and columns to update in order to change reserved words. The tables preceded by "*master.dbo.*" occur only in the *master* database. All other tables occur in *master* and in user database. Be certain you are using the correct database before you attempt the update by issuing a **select db_name()** query.

Type of Identifier	Table to Update	Column Name
user name	*sysusers*	*name*
login names	*master.dbo.syslogins*	*name*
segment names	*syssegments*	*name*
device name	*sysdevices*	*name*
remote server name	*sysservers*	*srvname*
remote server network name	*sysservers*	*srvnetname*
character set names	*master.dbo.syscharsets*	*name*
language name	*master.dbo.syslanguages*	*name*
	master.dbo.syslogins	*language*

Table 1-8: System Tables and Columns to Update When Changing Identifiers

The following table lists considerations and other changes that might be needed if you change the identifiers:

Identifier Type	Considerations
login name	Also change the user name in each database where this person is a user.
user name	Drop, edit, and recreate all procedures, triggers, and views that use qualified (*owner_name.object_name*) references to objects owned by this user. Change all applications and SQL source scripts that use qualified object names to use the new user name. Note that you do not have to drop the objects themselves; *sysusers* is linked to *sysobjects* by the column that stores the user's ID, not the user's name.
device name	Change any SQL source scripts or applications that reference the device name to use the new name.
remote server name	Also change the name on the remote server. If the name that **sp_checkreswords** reports is the name of the local server, you must reboot the server before you can issue or receive remote procedure calls.
remote server network name	Change the server's name in the interfaces files.
remote server login name	Also change the name on the remote server.

Table 1-9: Considerations When Changing Identifiers

Identifier Type	Considerations
segment name	Drop and recreate all procedures that create tables or indexes on the segment name. Change all applications that create objects on segments to use the new segment name.
character set name	None.
language name	Change both *master.dbo.syslanguages* and *master.dbo.syslogins*. The update to *syslogins* may involve many rows. Also change the names of your localization files.

Table 1-9: Considerations When Changing Identifiers (continued)

Using Delimited Identifiers

- You can use delimited identifiers for table names, column names, and view names. You cannot use delimited identifiers any other places where identifiers are needed.
- If you choose to use delimited identifiers, set the quoted identifier option on and drop and re-create all of the procedures, triggers and views that use the identifier. Edit the text for these objects, enclosing the reserved words in double quotes, and enclosing all character strings in single quotes. The syntax for the **set** command is:

  ```
  set quoted_identifiers on
  ```

  The following example shows the changes to make to queries in order to use delimited identifiers. This example updates a table named *work*, with columns named *key* and *level*. Here is the original query, which enclosed character literals in double quotes, and the edited version of the query for use with delimited identifiers:

  ```
  /* pre-release 10.0 version of query */
  update work set level = "novice"
      where key = "19-732"

  /* 10.0 version of query, using
  ** the quoted identifiers option
  */
  update "work" set "level" = 'novice'
      where "key" = '19-732'
  ```

- All applications that use the reserved word as an identifier must be changed as follows:
  - The application must set the quoted identifier option on.

- All uses of the reserved word must be enclosed in double quotes.
- All character literals that the application uses while the quoted identifier option is turned on must be enclosed in single quotes. Otherwise, SQL Server attempts to interpret them as object names.

For example, the following query results in an error message:

```
set quoted_identifier on

select * from titles where title_id like "BU%"
```

```
Msg 207, Level 16, State 2:
Server 'beta10', Line 1:
Invalid column name 'BU%'.
```

Here is the correct query:

```
select * from titles where title_id like 'BU%'
```

• Stored procedures that you create while the delimited identifiers are in effect can be run without turning on the option. (The **allow updates** option works this way, also.) This means that you can turn on quoted identifier mode, drop a stored procedure, edit it to insert quotation marks around reserved words used as identifiers, and re-create the procedure. All users can execute the procedure without using **set quoted_identifier** themselves.

Messages

• `Found no reserved words used as database object names.`

No tables, views, procedures, triggers, rules or defaults in the current database use reserved words as names.

• `Found no reserved words used as names for database-wide objects.`

No items such as segments or user names use reserved words as names.

• `No user with the specified name exists in the current database.`

The user name you specified is not a user in the current database. Be sure you spelled the name correctly, and be sure you are using the correct database.

Permissions

Any user can execute **sp_checkreswords**.

Tables Used

#uids, *master.dbo.spt_values*, *master.dbo.syscharsets*, *master.dbo.sysdatabases*, *master.dbo.sysdevices*, *master.dbo.syslanguages*, *master.dbo.syslogins*, *master.dbo.sysremotelogins*, *master.dbo.sysservers*, *master.dbo.sysmessages*, *syscolumns*, *sysindexes*, *sysobjects*, *syssegments*, *systypes*, *sysusers*

See Also

Commands	**reconfigure, set**
System procedures	**sp_configure, sp_depends, sp_rename, sp_renamedb**

sp_clearstats

Function

Initiates a new accounting period for all server users or for a specified user. Prints statistics for the previous period by executing **sp_reportstats**.

Syntax

`sp_clearstats [`*`user_name`*`]`

Parameters

user_name – is the user's login name.

Examples

1. **`sp_clearstats`**

```
Name    Since            CPU   Percent CPU    I/O   Percent I/O
------  -----------   ------  ------------  -----   -------------
probe   Jun 19 1990        0            0%      0        0%
julie   Jun 19 1990    10000      24.9962%   5000        24.325%
jason   Jun 19 1990    10002      25.0013%   5321        25.8866%
ken     Jun 19 1990    10001      24.9987%   5123        24.9234%
kathy   Jun 19 1990    10003      25.0038%   5111        24.865%
(5 rows affected)
```

```
Total CPU   Total I/O
---------       ---------
40006       20555
```

```
5 login accounts cleared.
```

Initiates a new accounting period for all users.

2. **`sp_clearstats kathy`**

```
Name  Since         CPU     Percent CPU     I/O     Percent I/O
----- -----------   -----   ------------    -----   -----------
KATHY Jul 24 1990   498     49.8998%        483924  9.1829%
(1 row affected)
```

```
Total CPU   Total I/O
---------   ----------
998         98392
1 login account cleared.
(1 row affected, return status = 0)
```

Initiates a new accounting period for the user "kathy."

Comments

- **sp_clearstats** creates an accounting period, and should be run only at the end of a period.
- **sp_clearstats** clears out the accounting statistics; the statistics should be recorded **before** running the procedure.
- **sp_clearstats** updates the *syslogins* field *accdate* and clears the *syslogins* fields *totcpu* and *totio*.

Messages

- `number login account(s) cleared.`

  The **sp_clearstats** command initiated a new accounting period for *number* users.

Permissions

Only a System Administrator can execute **sp_clearstats**.

Tables Used

master.dbo.syslogins, *sysobjects*

See Also

System procedures	**sp_reportstats**

sp_commonkey

Function

Defines a common key—columns that are frequently joined—between two tables or views.

Syntax

```
sp_commonkey tabaname, tabbname, col1a, col1b
    [, col2a, col2b, ..., col8a, col8b]
```

Parameters

tabaname – is the name of the first table or view to be joined.

tabbname – is the name of the second table or view to be joined.

col1a – is the name of the first column in table or view *tabaname* that makes up the common key. Specify at least one pair of columns (one column from the first table or view, and one from the second table or view).

The number of columns in each table or view must be the same, and their datatypes must be the same. Their lengths and nulltypes need not be the same. Up to eight columns from each table or view can participate in the common key.

col1b – is the name of the partner column in table or view *tabbname* that is joined with *col1a* in table or view *tabaname*.

Examples

1. `sp_commonkey projects, departments, empid, empid`

   Assume two tables, *projects* and *departments*, each with a column named *empid*. This statement defines a frequently used join on the two columns.

Comments

- Common keys are created in order to make explicit a logical relationship that is implicit in your database design. The information can be used by an application.
- Executing **sp_commonkey** adds the key to the *syskeys* system table. To display a report on the common keys that have been defined, execute **sp_helpkey**.

- You must be the owner of at least one of the two tables or views in order to define a common key between them.
- The number of columns from the first table or view must be the same as the number of columns from the second table or view. Up to eight columns from each table or view can participate in the common key. The datatypes of the common columns must also agree. For columns that take a length specification, the lengths can differ. The nulltypes of the common columns need not agree.
- The installation process runs **sp_commonkey** on appropriate columns of the system tables.

Messages

- `First table in the common key doesn’t exist.`

  The table or view you gave as *tabaname* doesn’t exist in the current database.
- `New common key added.`

  The common key between the specified tables or views has been added to *syskeys*.
- `Only the table owner may define its common keys.`

  You aren’t the owner of either *tabaname* or *tabbname*.
- `Second table in the common key doesn’t exist.`

  The table or view you gave as *tabbname* doesn’t exist in the current database.
- `Table or view name must be in current database.`

  Either the column pair that you specified doesn’t exist, or the columns in the pair are different types.
- `The tables have no such` *`nth`* `column or the columns are of different types.`

  Either the column pair that you specified doesn’t exist, or the columns in the pair are different types.

Permissions

Only the owner of *tabaname* or *tabbname* can issue **sp_commonkey**.

Tables Used

syscolumns, *syskeys*, *sysobjects*

See Also

Commands	**create trigger**
System procedures	**sp_dropkey, sp_foreignkey, sp_helpjoins, sp_helpkey, sp_primarykey**
Topics	Joins

sp_configure

Function

Displays or changes configuration variables.

Syntax

```
sp_configure [config_name [, config_value]]
```

Parameters

config_name – is the name of the configuration variable. SQL Server understands any unique string that is part of the configuration name.

config_value – is the value for the configuration option.

Examples

1. **sp_configure**

   Displays a list of all the configuration parameters with their current and permissible range of values.

   The report contains four columns:

   - The *minimum* column contains the minimum possible value for the variable.
   - The *maximum* column contains the maximum possible value.
   - The *config_value* column of the report contains the value to which the configuration variable has been set with **sp_configure**. It changes after you execute **sp_configure**. (This is the value in *sysconfigs.value*.)
   - The *run_value* column contains the value SQL Server is using. It changes after you run the **reconfigure** command, or, in some cases, after SQL Server is rebooted. (This is the value in *syscurconfigs.value*.)

2.
   ```
   sp_configure "recovery interval", 3
   ```

   Sets the system **recovery interval** to 3 minutes.

Comments

- With no parameters, **sp_configure** displays all of the possible options with their current settings and range of permitted values. If you specify **sp_configure** *config_name*, the value of *config_name* appears. Any user can execute **sp_configure** with no parameters or with one parameter.

- A System Administrator can execute **sp_configure** with two parameters to change the value of a specific configuration variable to a new *config_value* specified in the second parameter, except that only a System Security Officer can execute **sp_configure** with the **password expiration interval, audit queue size, allow updates**, and **remote access** options.
- Once **sp_configure** completes successfully, the System Administrator installs the changed *config_value* by:
  - Issuing the **reconfigure** command, then
  - Restarting SQL Server (for all non-dynamic variables).

  See the *System Administration Guide* for details.
- Use **reconfigure with override** when you set **allow updates** on, or when you set a configuration variable to a value that SQL Server considers less than optimal.
- To instruct SQL Server to supply a default configuration variable, give the value 0 as the *config_value*.

List of Configuration Variables

- The following briefly describes the configuration variables. For more information, see "Fine Tuning Performance and Operations" in the *System Administration Guide*.
  - **additional network memory** allocates additional memory for clients which request packet sizes that are larger than the default packet size for the server.
  - **allow updates** allows system tables to be updated directly. The default is 0 (off).
  - **audit queue size** determines the number of audit records that the audit queue can hold. The default is 100.
  - **cpu flush** specifies how many machine clock ticks to accumulate before adding cpu usage data to *syslogins* for use in chargeback accounting statistics.
  - **database size** sets the default number of megabytes allocated to each new user database. The default run value is 2 (megabytes).
  - **default character set id** is the number of the default character set used by the server.

- **default language** is the number of the language that is used to display system messages unless a user has chosen another language from those available on the server.
- **default network packet size** sets the default size of network packets for all users on SQL Server.
- **default sortorder id** is the number of the sort order that is the current default on this SQL Server. **Do not change this variable**. See the *System Administration Guide* for more information about changing the sort order.
- **devices** controls the number of database devices that SQL Server can use.
- **engine adjust interval** is not currently used.
- **extent i/o buffers** allocates the specified number of extents (8 data pages) for use by **create index**.
- **fillfactor** determines how full SQL Server makes each page when it is creating a new index on existing data (unless the user specifies some other value in the **create index** statement). The default run value is 0.
- **i/o flush** specifies how many disk I/Os to accumulate before flushing the data to *syslogins* for use in chargeback accounting.
- **identity burning set factor** determines the percentage of potential IDENTITY column values that is made available in each block. The default value, 5000, releases .05% of the potential IDENTITY column values for use at a time.
- **language in cache** is the maximum number of languages that can simultaneously be held in the language cache. The default is 3.
- **locks** sets the number of available locks. The default run value is 5000.
- **max online engines** controls the number of engines in a symmetric multiprocessor environment.
- **maximum network packet size** sets the maximum network packet size that a client program can request.
- **memory** sets the size of memory, in 2K units, that SQL Server allocates from the operating system. The default varies according to platform.
- **min online engines** is not currently used.

- **nested trigger** determines whether triggers can call other triggers (that is, be "nested") or not. The default is 1 (nested triggers enabled).
- **open databases** sets the maximum number of databases that can be open at one time on SQL Server. The default run value is 12.
- **open objects** sets the maximum number of database objects that can be open at one time on SQL Server. The default run value is 500.
- **password expiration interval** is the number of days that passwords remain in effect after they are changed. The default is 0 (passwords do not expire).
- **pre-read packets** controls the number of packets that a site handler will pre-read in connections with remote servers. The default is 3.
- **procedure cache** gives the percentage of memory allocated to the procedure cache after SQL Server's memory needs are met. The default run value is 20.
- **recovery flags** sets a toggle that determines what information SQL Server displays on the console during recovery. The default run value is 0, which means that SQL Server displays only the database name and a message saying that recovery is in progress.
- **recovery interval** sets the maximum number of minutes per database that SQL Server should use to complete its recovery procedures in case of a system failure. The default is 5 (minutes per database).
- **remote access** determines whether users from remote servers can access this SQL Server. The default is 1, to allow SQL Server to communicate with Backup Server.
- **remote connections** controls the limit on active connections initiated to and from this SQL Server. The default is 20.
- **remote logins** controls the number of active user connections from this SQL Server to remote servers. The default is 20.
- **remote sites** controls the number of simultaneous remote sites that can access this SQL Server. The default is 10.
- **stack size** sets the size of SQL Server's stack.
- **tape retention** sets the number of days that you expect to retain each tape after it has been used for a database or transaction log dump. The default run value is 0.

- **time slice** sets the number of milliseconds that SQL Server's scheduler allows a user process to run. The default run value is 100 milliseconds.
- **upgrade version** is changed by the upgrade program provided with new releases.
- **user connections** sets the maximum number of user connections that can be connected to SQL Server at the same time. The maximum value for your system is stored in the global variable *@@max_connections*, and varies according to platform and operating system.

Messages

- `Configuration option changed. Run the RECONFIGURE command to install.`

  After changing a configuration variable with **sp_configure**, the change does not take effect until the **reconfigure** command is issued and (for all but **allow updates** and **recovery interval**) SQL Server is restarted.

- `Configuration option doesn't exist.`

  The name supplied as the *config_name* parameter is unknown.

- `Configuration option is not unique.`

  The name supplied as the *config_name* parameter is not unique. No configuration variable was changed. For example, two of the configuration variables are **recovery interval** and **recovery flags**. Using *recovery* for the *config_name* parameter generates this message because it matches both names. The complete names that match the string supplied are printed out so you can see how to make the *config_name* more specific.

- `Configuration option value is not legal.`

  The *config_value* supplied is not in the range of permissible values for the specified configuration variable. For a display of the range of permissible values, re-run **sp_configure** with the name of the configuration variable as the only parameter.

  A *config_value* of 0 is always legal. It instructs SQL Server to set the configuration value to its default.

- `You can't set the number of devices to be less than the number of devices already defined in sysdevices.`

  Use **sp_helpdevice** to see a list of the devices defined for this server.

- `Can’t run sp_configure from within a transaction.`

  sp_configure modifies system tables, so it cannot not be run within a transaction.

- `You can’t set the default language to a language ID that is not defined in syslanguages.`

  Use **sp_helplanguage** to see the list of official language names available on this SQL Server.

- `Maximum file descriptors or FILLM process quota too low to support requested number of user connections. Configuration variable 'user connections' will not be modified.`

  Use this command:

  ```
  select @@max_connections
  ```

  to find the maximum value to which **user connections** can be configured.

Permissions

Any user can execute **sp_configure** with no parameters or only the first parameter (*optname*). A System Administrator can execute **sp_configure** with both parameters, except for the **password expiration interval, audit queue size, allow updates**, and **remote access** variables. Only a System Security Officer can set these variables.

Tables Used

master.dbo.spt_values, *master.dbo.sysdevices*, *master.dbo.sysservers*, *master.dbo.sysconfigures*, *master.dbo.syscurconfigs*, *master.dbo.sysdevices*, *master.dbo.syslanguages*, *master.dbo.sysmessages*, *master.dbo.sysservers*, *sysobjects*

See Also

Commands	reconfigure, set
System procedures	sp_addlanguage, sp_auditoption, sp_dboption, sp_droplanguage, sp_modifylogin

sp_cursorinfo

Function

Reports information about a specific cursor or all cursors that are active.

Syntax

sp_cursorinfo [{*cursor_level* | null}] [, *cursor_name*]

Parameters

cursor_level | **null** – is the level about which SQL Server returns information for the cursors. You can specify the following for *cursor_level*:

Level	Types of Cursors
N	Any cursors declared inside stored procedures at a specific procedure nesting level. You can specify any positive number for its level.
0	Any cursors declared outside stored procedures.
-1	Any cursors from either of the above. You can substitute any negative number for this level.

Table 1-10: Cursor Information Levels

If you want information about cursors with a specific *cursor_name*, regardless of cursor level, specify **null** for this parameter.

cursor_name – is the specific name for the cursor. SQL Server reports information about all active cursors which use this name at the *cursor_level* you specify. If you omit this parameter, SQL Server reports information about all the cursors at that level.

Examples

1. **`sp_cursorinfo 0, authors_crsr`**

```
Cursor name 'authors_crsr' is declared at nesting level '0'.
The cursor has been successfully opened 1 times.
The cursor is not open.
The cursor will remain open when a transaction is commited or
rolled back.
The number of rows returned for each FETCH is 1.
The cursor is updatable.
There are 3 columns returned by this cursor.
The result columns are:
Name = 'au_id', Table = 'authors', Type = ID,
     Length = 11 (updatable)
Name = 'au_lname', Table = 'authors', Type = VARCHAR,
Length = 40 (updatable)
Name = 'au_fname', Table = 'authors', Type = VARCHAR,
Length = 20 (updatable)
```

Displays the information about the cursor named *authors_crsr* at level 0.

2. **`sp_cursorinfo null, author_sales`**

```
Cursor name 'author_sales' is declared on procedure 'au_sales'.
Cursor name 'author_sales' is declared at nesting level '1'.
The cursor has been successfully opened 1 times.
The cursor is positioned after the last row.
The cursor will be closed when a transaction is commited or
rolled back.
The number of rows returned for each FETCH is 1.
The cursor is updatable.
There are 3 columns returned by this cursor.
The result columns are:
Name = 'title_id', Table = 'titleauthor', Type = ID,
     Length = 11 (updatable)
Name = 'title', Table = 'titles', Type = VARCHAR,
     Length = 80 (updatable)
Name = 'total_sales', Table = 'titles', Type = INT (updatable)
```

Displays the information about any cursors named *author_sales* declared by a user across all levels.

Comments

- If you do not specify either *cursor_level* or *cursor_name*, SQL Server displays information about all active cursors. Active cursors are those declared by a user and allocated by SQL Server.

- SQL Server reports the following information about each cursor:
  - The cursor name, its nesting level, and the procedure name if it is declared in a stored procedure.
  - The number of times the cursor has been opened.
  - Whether the cursor is open or closed. If the cursor is open, it indicates the current cursor position and the number of rows fetched.
  - Whether the open cursor will be closed if the cursor's current position is deleted.
  - Whether the cursor will remain open or be closed if the cursor's current transaction is committed or rolled back.
  - The number of rows returned for each fetch of that cursor.
  - Whether the cursor is updatable or read-only.
  - The number of columns returned by the cursor. For each column it displays the column name, the table name or expression result, and if it is updatable.

  In addition to the above, **sp_cursorinfo** displays the **showplan** output for the cursor. See the **set** command in Volume 1 of the *SQL Server Reference Manual* for more information about **showplan**. The output from **sp_cursorinfo** varies depending on the status of the cursor.

Messages

- `There are no active cursors.`

  SQL Server could not find any declared cursors.

- `There are no active cursors that match the search criteria.`

  SQL Server could not find any declared cursors that match the values you specified for *cursor_level* and *cursor_name*.

Permissions

Any user can execute **sp_cursorinfo.**

Tables Used

sysobjects

See Also

Commands	**declare cursor, set**
Topics	Cursors

sp_dboption

Function

Displays or changes database options.

Syntax

`sp_dboption [`*`dbname`*`, `*`optname`*`, {true | false}]`

Parameters

dbname – is the name of the database in which to set the option. You must be using *master* to execute **sp_dboption** with parameters (that is, in order to change a database option). You cannot, however, change *master*'s database option settings.

optname – is the name of the option to set or unset. SQL Server understands any unique string that is part of the option name. Use quotes around the option name if it is a keyword or includes embedded blanks or punctuation.

{**true** | **false**} – **true** to set the option, **false** to unset the option.

Examples

1. **`sp_dboption`**

   Displays a list of the database options:

```
Settable database options

 database_options
 ------------------------
 ALL SETTABLE OPTIONS
 abort tran on log full
 allow nulls by default
 auto identity
 dbo use only
 ddl in tran
 no chkpt on recovery
 no free space acctg
 read only
 select into/bulkcopy
 single user
 trunc log on chkpt
 trunc. log on chkpt.
```

2.
```
use master
go
sp_dboption pubs2, "read", true
use pubs2
go
checkpoint
go
```
   Makes the database *pubs2* read only. The **read** string uniquely identifies the **read only** option from among all available database options. Note the use of quotes around the keyword **read**.

3.
```
use master
go
sp_dboption pubs2, "read", false
use pubs2
go
checkpoint
go
```
   Makes the database *pubs2* writable again.

4.
```
use master
go
sp_dboption pubs2, "select into", true
go
checkpoint
go
```
   Allows **select into** and **bcp** operations on tables in the *pubs2* database. The **select into** string uniquely identifies the **select into/bulkcopy** option from among all available database options. Note that quotes are required around the option because of the embedded space.

5.
```
use master
go
sp_dboption mydb, “auto identity”, true
go
checkpoint
go
```
   Automatically defines 10-digit IDENTITY columns in new tables created in *mydb*. The IDENTITY column, *SYB_IDENTITY_COL*, is defined in each new table that is created without specifying either a **primary** key, a **unique** constraint, or an IDENTITY column. The column is not visible when you select all columns with the **select *** statement. To retrieve it, you must explicitly include the column name in the select list.

Comments

- The *master* database option settings cannot be changed.
- To display a list of the user-settable database options, execute **sp_dboption** with no parameters from inside the *master* database.
- For a report on which database options are set in a particular database, execute **sp_helpdb**.
- The Database Owner or System Administrator can set or unset particular database options for all new databases by executing **sp_dboption** on *model*.
- After **sp_dboption** has been executed, the change does not take effect until the **checkpoint** command is issued in the database for which the option was changed.

Database Options

- The **abort tran on log full** option determines the fate of a transaction that is running when the last-chance threshold is crossed in the log segment of the specified database. The default value is **false**, meaning that the transaction is suspended and is awakened only when space has been freed. If you change the setting to **true**, all user queries that need to write to the transaction log are killed until space in the log has been freed.
- Setting the **allow nulls by default** option to **true** changes the default value of a column from **not null** to **null**, in compliance with the ANSI standard. The Transact-SQL default value for a column is **not null**, meaning that null values are not allowed in a column unless **null** is specified in the column definition. **allow nulls by default true** reverses this.
- While the **auto identity** option is **true**, a 10-digit IDENTITY column is defined in each new table that is created without specifying either a **primary** key, a **unique** constraint, or an IDENTITY column. The column is not visible when you select all column with the **select *** statement. To retrieve it, you must explicitly mention the column name, *SYB_IDENTITY_COL*, in the select list.
- While the **dbo use only** option is set on (**true**), only the database's owner can use the database. When the **ddl in tran option** is set on (**true**), you can use certain data definition language commands in transactions. If **ddl in tran** is **true** in a particular database, commands such as **create table**, **grant**, and **alter table** are allowed inside transactions in that database. If **ddl in tran** is true in the *model*

database, the commands are allowed inside transactions in all databases created after **ddl in tran** was set in *model*.

◆ ***WARNING!***

Data definition language commands hold locks on system tables such as *sysobjects*. Avoid using them inside transactions; if you must use them, keep the transactions short.

Using any data definition language commands on *tempdb* within transactions may cause your system to grind to a halt. Always leave ddl in tran set to false in *tempdb*.

- The following commands can be used inside a user-defined transaction only if the **ddl in tran** option is set to **true**:

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

Table 1-11: DDL Commands Allowed in Transactions

- The following commands cannot be used inside a user-defined transaction under any circumstances:

alter database	**load database**	**truncate table**
create database	**load transaction**	**update statistics**
disk init	**reconfigure**	
drop database	**select into**	

Table 1-12: DDL Commands Not Allowed in Transactions

In addition, the following system procedures cannot be used inside user-defined transactions because they create temporary tables: **sp_helpdb**, **sp_helpdevice**, **sp_helpindex**, **sp_helpjoins**, **sp_helpserver**, and **sp_spaceused**.

System procedures that change the *master* database cannot be used inside user-defined transactions.

- The **no free space acctg** option suppresses free space accounting and execution of threshold actions for the non-log segments. This speeds recovery time because the free-space counts will not be recomputed for those segments.

- The **no chkpt on recovery** option is set on (**true**) when an up-to-date copy of a database is kept. In these situations, there is a "primary" and a "secondary" database. Initially, the primary database is dumped and loaded into the secondary database. Then, at intervals, the transaction log of the primary database is dumped and loaded into the secondary database.

  If this option is set off (**false**), the default condition, a checkpoint record is added to a database after it is recovered when you restart SQL Server. This checkpoint, which insures that the recovery mechanism won't be unnecessarily re-run, changes the sequence number and causes a subsequent load of the transaction log from the primary database to fail.

  Turning on this option for the secondary database causes it not to get a checkpoint from the recovery process, so that subsequent transaction log dumps from the primary database can be loaded into it.

- The **read only** option means that users can retrieve data from the database, but can't modify any data.

- Setting the **select into/bulkcopy** option on enables the use of **writetext**, **select into** a permanent table, or "fast" bulk copy into a table that has no indexes or triggers, using **bcp** or the bulk copy library routines. Because a transaction log dump cannot recover these unlogged operations, **dump transaction** to a dump device is prohibited. After non-logged operations are completed, turn **select into/bulk copy** off and issue **dump database**.

  Issuing the **dump transaction** statement after unlogged changes have been made to the database with **select into** or bulk copy produces an error message instructing you to use **dump database** instead. (The **writetext** command does not have this protection.)

  You do not have to set the **select into/bulkcopy** option on in order to **select into** a temporary table, since *tempdb* is never recovered. The option need not be on in order to run **bcp** on a table that has indexes, because tables with indexes are always copied with the slower version of bulk copy and are logged.

- When **single user** is set to **true**, only one user at a time can access the database.

- The **trunc log on chkpt** option means that the transaction log is truncated (committed transactions are removed) every time the **checkpoint** checking process occurs (usually more than once per minute). When the Database Owner runs **checkpoint** manually, however, the log is **not** truncated. It may be useful to turn this option on while doing development work, to prevent the log from growing.

While the **trunc log on chkpt** option is on, **dump transaction** to a dump device is prohibited, since dumps from the truncated transaction log cannot be used to recover from a media failure. Issuing the **dump transaction** statement produces an error message instructing you to use **dump database** instead.

See the *System Administration Guide* for additional information on database options.

Messages

- `Can’t run sp_dboption from within a transaction.`

  sp_dboption modifies system tables, so it cannot be run within a transaction.

- `Database option '`*`option_name`*`' turned [OFF | ON] for database '`*`database_name`*`'.`

  The **sp_dboption** command succeeded. This message reports on the option you have just set.

- `Database option doesn’t exist or can’t be set by user.`

  Either the option does not exist or the user does not have permission to set or unset it. Run **sp_dboption** with no parameters to display a list of the user-settable options.

- `Database option is not unique.`

  The name supplied as the *optname* parameter is not unique. No database option value was changed. For example, two of the database options are **dbo use only** and **read only.** Using **only** for the *optname* parameter generates this message because it matches both names. The complete names that match the string supplied are printed out so you can see how to make the *optname* more specific.

- `No such database—run sp_helpdb to list databases.`

  No database with the supplied name exists. Run **sp_helpdb** to get a list of databases.

- `Run the CHECKPOINT command in the database that was changed.`

  The change in the database option takes effect only after the **checkpoint** command is run.

- `Settable database options.`

  Executing **sp_dboption** with no parameters displays a list of the user-settable options.

- `The database is currently in use -- 'read only' option disallowed.`

  You must wait until no one is using the database before issuing this command. Use **sp_who** to monitor usage.

- `The 'master' database’s options cannot be changed.`

  No one can change any of the *master*’s database option settings.

- `Usage: sp_dboption [dbname, optname, {true | false}]`

  Either the *optname* parameter was omitted or the third parameter was something other than TRUE or FALSE.

- `You must be in the 'master' database in order to change database options.`

  In order to change a database option (of any database other than *master*), execute the **sp_dboption** procedure, with the appropriate parameters, while using *master*.

- `Run sp_dboption with no parameters to see options.`

  The command failed. Check the spelling of the options and reissue **sp_dboption**.

Permissions

Any user can execute **sp_dboption** with no parameters (display options only). Only a System Administrator or the Database Owner can execute **sp_dboption** with parameters (change an option).

Tables Used

master.dbo.spt_values, *master.dbo.sysdatabases*, *master.dbo.sysmessages*, *master.dbo.sysprocesses*, *sysobjects*

See Also

Commands	**checkpoint, select**
System procedures	**sp_configure, sp_helpdb, sp_helpjoins**

sp_dbremap

Function

Forces SQL Server to recognize changes made by **alter database**. Run this procedure only if instructed to do so by SQL Server messages.

Syntax

```
sp_dbremap database_name
```

Parameters

database_name – is the name of the database in which the **alter database** command was interrupted.

Examples

1. **`sp_dbremap sample_db`**

   An **alter database** command changed the database *sample_db*. This command makes the changes visible to SQL Server.

Comments

- If an **alter database** statement issued on a database that is in the process of being dumped is interrupted, SQL Server prints a message instructing the user to execute **sp_dbremap**.

  Any changes to *sysusages* during a database or transaction dump are not copied into active memory until the dump completes to ensure that database mapping does not change during the dump. Running **alter database** makes changes to system tables on the disk immediately. In-memory allocations cannot be changed until a dump completes. This is why **alter database** pauses.

  When you execute **sp_dbremap**, it must wait until the dump process completes.

- If you are instructed to run **sp_dbremap**, but do not do it, the space you have allocated with **alter database** does not become available to SQL Server until the next reboot.

Messages

- `Can’t run sp_dbremap from within a transaction.`

  sp_dbremap modifies system tables, so it cannot be run within a transaction.

- `'`*`database_name`*`' is not a valid identifier.`

  The database name you specified is not a valid identifier.
- `The specified database does not exist`

  The database name you specified is not the name of a database on this server.

Permissions

Only a System Administrator can execute **sp_remap**.

Tables Used

master.dbo.sysdatabases, sysobjects

See Also

Commands	**alter database, dump database, dump transaction**

sp_depends

Function

Displays information about database object dependencies—the view(s), trigger(s), and procedure(s) that depend on a specified table or view, and the table(s) and view(s) that the specified view, trigger, or procedure depends on.

Syntax

sp_depends *objname*

Parameters

objname – is the name of the table, view, stored procedure, or trigger to examine for dependencies. You cannot specify a database name. Use owner names if the object owner is not the user running the command and not the *dbo*.

Examples

1. **sp_depends sysobjects**

   Lists the database objects that depend on the table *sysobjects*.

2. **sp_depends titleview**

   ```
   Things that the object references in the current
   database.

   object           type        updated selected
   --------------   ----------- ------- -----
   dbo.authors      user table  no      no
   dbo.titleauthor  user table  no      no
   dbo.titles       user table  no      no

   Things inside the current database that reference
   the object.

   object           type
   ------------     ---------------
   dbo.tview2       view
   ```

3. **sp_depends “mary.titles”**

   Lists the database objects that depends on the *titles* table owned by the user *mary*. The quotes are needed, since “.” is a special character.

Comments

- Executing **sp_depends** lists all the objects, if any, that depend on *objname*, and all the objects, if any, that *objname* depends on. For example, views depend on one or more tables and can have procedures or other views that depend on them. An object that references another object is considered dependent on that object. References to objects outside the current database are not reported.
- The **sp_depends** procedure determines the dependencies by looking at the *sysdepends* table.
- The *updated* and *selected* columns in the report from **sp_depends** are meaningful if the object being reported on is a stored procedure or trigger. The values in these columns indicate whether the stored procedure or trigger updates or selects from that object.
- **sp_depends** follows SQL Server's rule for finding objects:
  - If the user doesn't specify an owner name, and the user executing the command owns an object with the specified name, that object is used.
  - If the user doesn't specify an owner name, and the user does not own an object of that name, but the *dbo* does, the *dbo*'s object is used.
  - If neither the user nor the *dbo* owns an object of that name, the command reports an error condition, even if an object exists in the database with that object name, but different owner.
  - If the user and the *dbo* both own objects with the specified name, and the user wants to access the *dbo*'s object, the name must be specified, as in *dbo.objectname*.
- Objects owned by database users other than the user executing a command and the *dbo* must always be qualified with the owner's name, as in Example 3.

Messages

- `Object does not exist in this database.`

  The object name supplied for the *objname* parameter does not exist in the current database.
- `Object doesn't reference any object and no objects reference it.`

  Nothing depends upon *objname* and *objname* doesn't reference any objects.

- `Object must be in the current database.`

  You cannot reference an object that is not in your current database.

- `Things inside the current database that reference the object.`

  These are the objects in the current database that reference *objname*. (See Example 2 on page 1-124.)

- `Things the object references in the current database.`

  These are the objects in the current database that *objname* depends on. (See Example 2 on page 1-124.)

Permissions

All users can execute **sp_depends**.

Tables Used

master.dbo.spt_values, *master.dbo.sysmessages*, *sysdepends*, *sysobjects*, *sysusers*

See Also

Commands	**create procedure, create table, create view, execute**
System procedures	**sp_help**

sp_diskdefault

Function

Sets a database device's status to **defaulton** or **defaultoff**. This indicates whether or not a database device can be used for database storage if the user does not specify a database device or specifies **default** with the **create database** or **alter database** commands.

Syntax

```
sp_diskdefault logical_name {defaulton | defaultoff}
```

Parameters

logical_name – is the logical name of the device as given in *master.dbo.sysdevices.name*. The device must be a database device rather than a dump device.

defaulton | **defaultoff** – **defaulton** if the specified database device is to be designated a default database device; **defaultoff** if the specified database device is not to be designated a default database device.

The keyword **defaulton** is most often used after a database device is added to the system with **disk init**. The keyword **defaultoff** is most often used to change the default status of the *master* device (which is on when SQL Server is first installed).

Examples

1. `sp_diskdefault master, defaultoff`

   The *master* device is no longer used by **create database** or **alter database** for default storage of a database.

Comments

- A default database device is one that is used for database storage by **create database** or **alter database** if the user does not specify a database device name or specifies the keyword **default**.
- You can have multiple default devices. They are used in the order they appear in the *master.dbo.sysdevices* table (that is, alphabetical order). When the first default device is filled, the second default device is used, and so on.
- When you first install SQL Server, the master device, *d_master*, is the only default database device.

➤ ***Note***

Once you initialize devices to store user databases, use **sp_diskdefault** to turn off the master device's default status. This prevents users from accidentally creating databases on the master device, and makes recovery of the *master* database simpler.

- To find out which database devices are default database devices, execute **sp_helpdevice**.

Messages

- `Can't run sp_diskdefault from within a transaction.`

  sp_diskdefault modifies system tables, so it cannot be run within a transaction.

- `No such device exists -- run sp_helpdevice to list the SQL Server devices.`

  The device name supplied for the *logicalname* parameter doesn't exist. Run **sp_helpdevice** without a parameter to see a list of all devices. To add a new database device to the system, use the **disk init** command.

- `The device name supplied is not a database disk.`

  The device name supplied for the *device_name* parameter is in *sysdevices*, but it is a dump device rather than a database device. Run **sp_helpdevice** without a parameter to see a list of all devices. To add a new database device to the system, use the **disk init** command.

- `Usage: sp_diskdefault logicalname {defaulton | defaultoff}.`

  The second parameter must be either **defaulton** or **defaultoff**.

Permissions

Only a System Administrator can execute **sp_diskdefault.**

Tables Used

master.dbo.sysdevices, *sysobjects*

See Also

Commands	**alter database, create database, disk init**
System procedures	**sp_helpdevice**

sp_displaylogin

Function

Displays information about a login account.

Syntax

```
sp_displaylogin [login_name]
```

Parameters

login_name – is the user login account about which you want information if it is other than your own. You must be a System Security Officer or System Administrator to get information about someone else’s login account.

Examples

1. **`sp_displaylogin`**

   Displays information about your server login account.

2. **`sp_displaylogin bob`**

   Displays information about the login account “bob”. The information displayed depends on the role of the user executing **sp_displaylogin**.

Comments

- **sp_displaylogin** displays configured roles, so that even if you have made a role inactive with the **set** command, it is displayed.
- When you use **sp_displaylogin** to get information about your own account you do not need to use the *login_name* parameter. **sp_displaylogin** displays your server user ID, login name, full name, any roles that have been granted to you, date of last password change, and whether your account is locked.
- If you are a System Security Officer or System Administrator, you can use the *login_name* parameter to access information about any account.

Messages

- `No login with the specified name exists.`

  You specified an incorrect *login_name*.

Permissions

Any user can execute **sp_displaylogin** to get information about his or her own login account. System Security Officers and System Administrators can use **sp_displaylogin** with the *login_name* parameter to get information about other users' login accounts.

Tables Used

master.dbo.sysloginroles, *master.dbo.syslogins*, *master.dbo.syssrvroles*, *sysobjects*

See Also

Topics	Roles

sp_dropalias

Function

Removes the alias user name identity established with **sp_addalias**.

Syntax

```
sp_dropalias login_name
```

Parameters

login_name – is the name (in *master.dbo.syslogins*) of the user who was aliased to another user.

Examples

1. `sp_dropalias victoria`

   Assuming that "victoria" was aliased (for example, to the Database Owner) in the current database, this statement drops "victoria" as an aliased user from the database.

Comments

- Executing the **sp_dropalias** procedure deletes an alternate *suid* mapping for a user from the *sysalternates* table.
- When a user's alias is dropped, he or she no longer has access to the database for which the alias was created.

Messages

- `Alias user dropped.`

  The user is no longer aliased to another user in the current database. The user cannot use the database until reinstated by the Database Owner with **sp_adduser** or **sp_addalias**.
- `No alias for specified user exists.`

  The named user doesn't have an alias in the current database.
- `No login with the specified name exists.`

  The *login_name* you supplied has no account on SQL Server. No action was taken.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropalias**.

Tables Used

sysalternates, sysobjects

See Also

Commands	**use**
System procedures	**sp_addalias, sp_adduser, sp_changedbowner, sp_droplogin, sp_dropuser, sp_helpuser**

sp_dropdevice

Function

Drops a SQL Server database device or dump device.

Syntax

```
sp_dropdevice device_name
```

Parameters

device_name – is the name of the device as listed in *master.dbo.sysdevices.name*.

Examples

1. **`sp_dropdevice tape5`**

   Drops the device named *tape5* from SQL Server.

2. **`sp_dropdevice fredsdata`**

   Drops the database device named *fredsdata* from SQL Server. The device must not be in use by any databases.

Comments

- The **sp_dropdevice** procedure drops a device from SQL Server, deleting the device entry from *master.dbo.sysdevices*.
- **sp_dropdevice** does not remove a file that is being dropped as a database device; it makes the file inaccessible to SQL Server. Use operating system commands to delete a file after using **sp_dropdevice**.

◆ ***WARNING!***

You must restart SQL Server after you drop a device because the kernel has a process that is accessing the dropped device, and there is no way to kill the process. Restarting with startserver or dataserver frees up the logical device number.

Messages

- `Can’t run sp_dropdevice from within a transaction.`

  sp_dropdevice modifies system tables, so it cannot be run within a transaction.

- `Device dropped.`

  The device was dropped from the *master.dbo.sysdevices* table.

- `Device is being used by a database. You can’t drop it.`

  Only database devices that are not in use can be dropped. You must drop all the databases associated with the device before dropping the device.

- `No such device exists -- run sp_helpdevice to list the SQL Server devices.`

  You tried to drop a device that doesn’t exist on SQL Server.

Permissions

Only a System Administrator can execute **sp_dropdevice**.

Tables Used

master.dbo.sysdatabases, *master.dbo.sysdevices*, *master.dbo.sysusages*, *sysobjects*

See Also

Commands	**drop database**
System procedures	**sp_addumpdevice, sp_helpdb, sp_helpdevice**

sp_dropgroup

Function

Drops a group from a database.

Syntax

`sp_dropgroup` ***`grpname`***

Parameters

grpname – is the name of a group in the current database.

Examples

1.
```
sp_changegroup accounting, martha
sp_changegroup "public", george
sp_dropgroup purchasing
```

   The "purchasing" group has merged with the "accounting" group. These commands move "martha" and "george", members of the "purchasing" group, to other groups before dropping the group. The group name "public" is quoted because "public" is a reserved word.

Comments

- Executing **sp_dropgroup** drops a group name from a database's *sysusers* table.
- You cannot drop a group if it has members. You must execute **sp_changegroup** for each member before you can drop the group.

Messages

- `Can't drop the group 'public'.`

  The "public" group exists in every database. It is the group that all users belong to by default, and cannot be dropped.
- `Group has been dropped.`

  The command succeeded. The group no longer exists in the current database.
- `Group has members. It must be empty before it can be dropped.`

  Groups with members cannot be dropped. Reassign the members of the group to another group using **sp_changegroup**. A list of the group members appears after this message.

- `No group with the specified name exists.`

  The specified group doesn't exist.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropgroup**.

Tables Used

master.dbo.syssrvroles, *sysobjects*, *sysprotects*, *sysusers*

See Also

Commands	**grant, revoke, use**
System procedures	**sp_addgroup, sp_adduser, sp_changegroup, sp_dropuser, sp_helpgroup**

sp_dropkey

Function

Removes from the *syskeys* table a key that had been defined using **sp_primarykey**, **sp_foreignkey**, or **sp_commonkey**.

Syntax

sp_dropkey *keytype*, *tabaname* [, *tabbname*]

Parameters

keytype – is the type of key to drop. The *keytype* must be **primary**, **foreign**, or **common**.

tabaname – is the name of the key table or view that contains the key to drop.

tabbname – specifies the name of the second table in the relation if the *keytype* is **foreign** or **common**. If the *keytype* is **primary**, this parameter is not needed, since **primary** keys have no dependent tables. If the *keytype* is **foreign**, this is the name of the primary key table. If the *keytype* is **common**, give the two table names in the order in which they appear with **sp_helpkey**.

Examples

1. **sp_dropkey primary, employees**

   Drops the primary key for the table *employees*. Any foreign keys that were dependent on the primary key for *employees* are also dropped.

2. **sp_dropkey common, employees, projects**

   Drops the common keys between the tables *employees* and *projects*.

3. **sp_dropkey foreign, titleauthor, titles**

   Drops the foreign key between the tables *titleauthor* and *titles*.

Comments

- Executing **sp_dropkey** deletes the specified key from *syskeys*. Only the owner of a table may drop a key on that table.
- Keys are created to make explicit a logical relationship that is implicit in your database design. This information can be used by an application program.

- Dropping a primary key automatically drops any foreign keys associated with it. Dropping a foreign key has no effect on a primary key specified on that table.
- Executing **sp_commonkey, sp_primarykey**, or **sp_foreignkey** adds the key to the *syskeys* system table. To display a report on the keys that have been defined, execute **sp_helpkey.**

Messages

- `Common keys dropped.`

  The **sp_dropkey** command succeeded, dropping the common keys and deleting them from *syskeys*.
- `Dependent foreign keys were also dropped.`

  When a primary key is dropped, any foreign keys that depend on it are also dropped.
- `Foreign key dropped.`

  The **sp_dropkey** command succeeded, dropping the foreign key and deleting it from *syskeys*.
- `No common keys exist between the two tables or views supplied.`

  There are no common keys between the *tabaname* and *tabbname* tables, or the table names were given in the wrong order. No action was taken. Use **sp_helpkey** to see the keys and the order in which to give the arguments.
- `No foreign key for the table or view exists.`

  tabaname has no foreign key defined.
- `No primary key for the table or view exists.`

  tabaname has no primary key defined.
- `Primary key for the table or view dropped.`

  The **sp_dropkey** command succeeded, dropping the primary key and deleting it from *syskeys*.
- `Table or view name must be in current database.`

  You can’t drop keys on tables or views in other databases.
- `The dependent table or view doesn’t exist in the current database.`

  The name supplied for the *tabbname* parameter isn’t a table or view in the current database.

- `The table or view named doesn’t exist in the current database.`

  The *tabaname* supplied isn’t a table or view in the current database.

- `Usage: sp_dropkey {primary | foreign | common},` *`tabaname`* `[,` *`tabbname`*`]. Type must be 'primary', 'foreign', or 'common'.`

  The *keytype* parameter should specify the type of key to drop.

- `You must be the owner of the table or view to drop its key.`

  You aren’t the owner of the table, so you can’t drop the key.

- `You must supply the dependent table or view as the third parameter.`

  When dropping a foreign or common key, both the *tabaname* and *tabbname* tables must be named.

Permissions

Only the owner of *tabaname* can issue **sp_dropkey**.

Tables Used

syskeys, *sysobjects*

See Also

System procedures	**sp_commonkey, sp_foreignkey, sp_helpkey, sp_primarykey**

sp_droplanguage

Function

Drops an alternate language from the server and removes its row from *master.dbo.syslanguages*.

Syntax

```
sp_droplanguage language [, dropmessages]
```

Parameters

language – is the official name of the language to drop.

dropmessages – drops all SQL Server system messages in *language*. You cannot drop a language with associated system messages without also dropping its messages by entering **dropmessages**.

Examples

1. **`sp_droplanguage french`**

   This command drops French from the set of available alternate languages, if there are no associated messages.

2. **`sp_droplanguage french, dropmessages`**

   This command drops French from the set of available alternate languages, if there are associated messages.

Comments

- Executing **sp_droplanguage** drops a language from a list of alternate languages by deleting its entry from the *master.dbo.syslanguages* table.
- If you try to drop a language that has system messages, the request fails unless you supply the **dropmessages** parameter.

Messages

- `language is not an official language name from syslanguages.`

  Use **sp_helplanguage** to see the list of official languages available on this SQL Server.

- `Can’t drop '`*`language`*`' because there are associated entries in master.dbo.sysmessages. Run sp_droplanguage with 'dropmessages' flag.`

  You cannot drop a language for which the *master* database contains associated system messages. Rerun **sp_droplanguage** with the **dropmessages** option to drop the language and all associated system messages.

- `The only legal value for the second parameter is 'dropmessages'.`

  You cannot specify any option other than **dropmessages**.

- `Language deleted.`

  The language is deleted from *master.dbo.syslanguages*. Error messages associated with this language are deleted from *master.dbo.sysmessages*.

Permissions

Only a System Administrator can issue **sp_droplanguage**.

Tables Used

master.dbo.syslanguages, *master.dbo.sysmessages*, *sysobjects*

See Also

System procedures	**sp_addlanguage, sp_helplanguage**

sp_droplogin

Function

Drops a SQL Server user login by deleting the user's entry in *master.dbo.syslogins*.

Syntax

```
sp_droplogin login_name
```

Parameters

login_name – is the name of the user as listed in *master.dbo.syslogins*.

Examples

1. **`sp_droplogin victoria`**

   Drops "victoria" from SQL Server.

Comments

- Executing **sp_droplogin** drops a user login from SQL Server, deleting the user's entry from *master.dbo.syslogins*.
- SQL Server reuses a dropped login's server user IDs, which compromises accountability. You may avoid dropping accounts at all and instead use **sp_locklogin** to lock any accounts that will no longer be used. If you do need to drop logins, be sure to audit these events (using **sp_auditsproc**) so that you have a record of them.
- **sp_droplogin** fails if the login to be dropped is a user in any database on the server. Use **sp_dropuser** to drop the user from a database. You cannot drop a user from a database if that user owns any objects in the database.
- If the login to be dropped is a System Security Officer, **sp_droplogin** verifies that at least one other unlocked System Security Officer's account exists. If not, **sp_droplogin** fails. Similarly, **sp_droplogin** ensures that there is always at least one unlocked System Administrator's account.

Messages

- `Can't run sp_droplogin from within a transaction.`

  sp_droplogin modifies system tables, so it cannot be run within a transaction.

- `Login dropped.`

  The user's entry in *master.dbo.syslogins* has been deleted. The user no longer has access to SQL Server.

- `No such account -- nothing changed.`

  The specified login name does not exist.

- `User exists or is an alias in at least one database. Drop user/alias before dropping login.`

  You cannot drop a login who is a user in any database on the server, or an user who has an alias in a database. Use **sp_dropuser** to drop a user from a database or **sp_dropalias** to drop the alias from the databases.

- `Warning: the specified account is currently active. Nothing changed.`

  You cannot drop an account if it is active. Run the command again when the user has logged off. You may be able to use **kill** to end their SQL Server session.

Permissions

Only the System Administrator can execute **sp_droplogin.**

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, sysobjects

See Also

System procedures	**sp_addlogin, sp_auditsproc, sp_changedbowner, sp_dropalias, sp_dropuser, sp_helpuser, sp_locklogin**
Topics	Login Management

sp_dropmessage

Function

Drops user-defined messages from *sysusermessages*.

Syntax

```
sp_dropmessage message_number [, language]
```

Parameters

message_number – is the message number of the message to drop. Message numbers must have a value of 20000 or higher.

language – is the language of the message to drop.

Examples

1. **`sp_dropmessage 20002, french`**

   Removes the French version of the message with the number 20002 from *sysusermessages*.

Comments

- The *language* parameter is optional. If included, only the message with the indicated *message_number* in the indicated language is dropped. If you do not specify a *language*, all messages with the indicated *message_number* are dropped.

Messages

- `language is not an official language name from syslanguages.`

  The *language* given is not a valid name in the *syslanguages* table.

- `Message number must be at least 20000.`

  Only user-defined messages, which have message numbers of 20000 or higher, can be deleted.

- `Message number message_number does not exist.`

  No message with the given message number exists in *sysusermessages*.

- `Message number` *`message_number`* `does not exist in the` *`language`* `language.`

  A message with the given message number does not exist in the *language* given.

- `Message deleted.`

  The message has been dropped.

- `User` *`user_name`* `does not have permission to drop message number` *`message_number`*`.`

- `User` *`user_name`* `does not have permission to drop message number` *`message_number`* `in the` *`language`* `language.`

  Only System Administrators, the Database Owner, and the user who originally created the message being dropped can delete a message.

Permissions

Only a System Administrator, the Database Owner, and the user who originally created the message being dropped can execute **sp_dropmessage**.

Tables Used

master.dbo.syslanguages, *sysobjects*, *sysusermessages*

See Also

System procedures	**sp_addmessage, sp_getmessage**

sp_dropremotelogin

Function

Drops a remote user login.

Syntax

```
sp_dropremotelogin remoteserver [, login_name
   [, remotename] ]
```

Parameters

remoteserver – is the name of the server which has the remote login to be dropped.

login_name – is the local server's user name that is associated with the remote server in the *sysremotelogins* table.

remotename – is the remote user name that gets mapped to *login_name* when logging in from the remote server.

Examples

1. **`sp_dropremotelogin GATEWAY`**

   Drops the entry for the remote server named GATEWAY.

2. **`sp_dropremotelogin GATEWAY, churchy`**

   Drops the entry for mapping remote logins from the remote server GATEWAY to the local user named "churchy".

3. **`sp_dropremotelogin GATEWAY, churchy, pogo`**

   Drops the login for the remote user "pogo" on the remote server GATEWAY that was mapped to the local user named "churchy".

Comments

- Executing **sp_dropremotelogin** drops a user login from a remote server, deleting the user's entry from *master.dbo.sysremotelogins*.
- For a more complete discussion on remote logins, see **sp_addremotelogin**.
- To add and drop local server users, use the system procedures **sp_addlogin** and **sp_droplogin**.

Messages

- `Can’t run sp_dropremotelogin from within a transaction.`

  sp_dropremotelogin modifies system tables, so it cannot be run within a transaction.

- `Remote login dropped.`

  The remote user’s entry in *master.dbo.sysremotelogins* has been deleted. The remote user no longer has access to this server.

- `There is no remote user '`*`remotename`*`' mapped to local user '`*`login_name`*`' from the remote server '`*`remoteserver`*`'.`

  The specified remote login name does not exist for the named server.

Permissions

Only the System Administrator can execute **sp_dropremotelogin**.

Tables Used

master.dbo.sysremotelogins, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addlogin, sp_addremotelogin, sp_addserver, sp_droplogin, sp_helpremotelogin, sp_helpserver**

sp_dropsegment

Function

Drops a segment from a database or unmaps a segment from a particular database device.

Syntax

sp_dropsegment *segname*, *dbname* [, *devname*]

Parameters

segname – is the name of the segment to drop.

dbname – is the name of the database.

devname – is the name of the database device for the segment *segname* to no longer use. This parameter is optional, except when dropping the system segments *system*, *default* and *logsegment* from a database device.

Examples

1. **sp_dropsegment indexes, pubs2**

   This command drops the segment *indexes* from the *pubs2* database.

2. **sp_dropsegment indexes, pubs2, dev1**

   This command unmaps the segment *indexes* from the database device *dev1*.

Comments

- You can drop a segment if it is not referenced by any table or index in the specified database.
- If you do not supply the optional argument *devname*, the segment is dropped from the specified database. If you do supply a *devname* name, the segment is no longer mapped to the named database device, but the segment is not dropped.
- Dropping a segment drops all thresholds associated with that segment.
- When you unmap a segment from one or more devices, SQL Server drops any thresholds that exceed the total space on the segment. When you unmap the *logsegment* from one or more devices, SQL Server recalculates the last-chance threshold.

- Using **sp_placeobject** to change future space allocations for a table or index from one segment to another dereferences the original segment. You can drop the original segment name with **sp_dropsegment**.
- For the system segments *system*, *default* and *logsegment*, you must specify the device name from which you want the segments dropped.

Messages

- `Can’t drop the '`*`egname`*`' segment completely.`

  You did not specify the device from which you want the segment dropped.
- `Can’t run sp_dropsegment from within a transaction.`

  sp_dropsegment modifies system tables, so it cannot be run within a transaction.
- `Segment dropped.`

  The procedure was successful. There is no longer a segment named *segname* in the specified database.
- `Segment reference to device dropped.`

  The procedure was successful. The segment *segname* no longer refers to database device *devname*.
- `Segment '`*`segname`*`' does not reference device '`*`devname`*`'.`

  The segment you tried to drop from *devname* isn’t referenced by *segname*. Run **sp_helpsegment** *segname* to list the devices that *segname* references.
- `The specified device is not used by the database.`

  The specified database doesn’t use device *devname*. Use **sp_helpsegment** to see which devices are referenced by *segname*.
- `The segment '`*`segname`*`' is being used.`

  You can’t drop a segment that is referenced by a table or index. If you still want to drop the segment, you must redefine the segment for the affected tables or indexes by using the system procedure **sp_placeobject**.
- `There is no such segment as '`*`segname`*`'.`

  The segment you have tried to drop does not exist. All segments for a database are listed in the *syssegments* table.

- `There is only one device mapping for the segment '`*`segname`*`' -- use sp_dropsegment with no device argument.`

  The *device* you have tried to drop is the last device reference for *segname*. It's illegal to drop the last device reference for a segment.

- `WARNING: There are no longer any segments referencing device '`*`devname`*`'. This device will no longer be used for space allocation.`

  The procedure was successful, but the device is now unassigned and can't be used for storing data or log information.

- `WARNING: There are no longer any segments referencing devices '`*`devname_list`*`'. These devices will no longer be used for space allocation.`

  The procedure was successful, but the devices are now unassigned and can't be used for storing data or log information.

- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use `*`database_name`*`' and try again.`

  sp_dropsegment can drop segments only in the database you are currently using. Issue the **use** command to open the database in which you want to drop a segment. Then run **sp_dropsegment** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropsegment**.

Tables Used

#temptable, *master.dbo.spt_values*, *sysdatabases*, *sysdevices*, *sysindexes*, *sysobjects*, *syssegments*, *systhresholds*, *sysusages*

See Also

System procedures	**sp_addsegment, sp_addthreshold, sp_helpsegment, sp_helpthreshold, sp_placeobject**

sp_dropserver

Function

Drops a server from the list of known servers.

Syntax

```
sp_dropserver server [, droplogins]
```

Parameters

server – is the name of the server to be dropped.

droplogins – indicates that any remote logins for *server* should also be dropped.

Examples

1. `sp_dropserver GATEWAY`

   This command drops the remote server GATEWAY.

Comments

- Executing **sp_dropserver** drops a server from a list of known servers, deleting the entry from the *master.dbo.sysservers* table.
- Running **sp_dropserver** on a server that has associated entries in the *master.dbo.sysremotelogins* table results in an error message stating that you must drop the remote users before you can drop the server. To drop all the remote logins for a server when dropping the server, supply the value **droplogins**.

Messages

- `Can’t run sp_dropserver from within a transaction.`

  sp_dropserver modifies system tables, so it cannot be run within a transaction.

- `Remote logins for remote server 'server' have been dropped.`

  The **sp_dropserver** command succeeded, and dropped the remote server and all the associated logins.

- `Server dropped.`

  The procedure was successful. The server named *server* is no longer accessible through this server and it can no longer access this server.

- `There are still remote logins for the server 'server'.`

  The server you want to drop has associated entries in the *sysremotelogins* table. You must either drop the remote logins with **sp_dropremotelogin** or use the **droplogins** parameter to the **sp_dropserver** system procedure.

- `There is not a server named 'server'.`

  The server you have tried to drop is not a known server. All known servers for a SQL Server are listed in the *master.dbo.sysservers* table.

- `Usage: sp_dropserver server [, droplogins]`

  The only valid parameter to **sp_dropserver** is **droplogins**.

Permissions

Only a System Security Officer can execute **sp_dropserver**.

Tables Used

master.dbo.sysremotelogins, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addserver, sp_dropremotelogin, sp_helpremotelogin, sp_helpserver**

sp_dropthreshold

Function

Removes a free-space threshold from a segment.

Syntax

```
sp_dropthreshold database, segment, free_pages
```

Parameters

database – is the database from which you are dropping the threshold. This must be the name of the current database.

segment – is the segment whose free space is monitored by the threshold. Use quotes when specifying the “default” segment.

free_pages – is the number of free pages at which the threshold is crossed.

Examples

1. **`sp_dropthreshold mydb, segment1, 200`**

   Removes a threshold from *segment1* of *mydb*. You must specify the database, segment, and amount of free space to identify the threshold.

Comments

- You cannot drop the last-chance threshold from the log segment.
- You can use the **no free space acctg** option of **sp_dboption** as an alternative to **sp_dropthreshold**. This option disables free-space accounting on non-log segments. You cannot disable free-space accounting on log segments.

Messages

- `Dropping threshold for segment 'segment_name' at 'pageno' pages.`

  The **sp_dropthreshold** command succeeded.

- `Segment 'segment_name' does not have a threshold at 'pageno' pages.`

  Run **sp_helpthreshold** to see the names of the thresholds in the current database.

- `Table '`*`systhresholds`*`' does not exist in database '`*`dbname`*`' -- cannot drop thresholds.`

  The *systhresholds* table is missing. This table is created when the database is created (or an upgrade to Release 10 is performed), and must not be removed.

- `There is no segment named '`*`segment_name`*`'.`

  Run **sp_helpsegment** to see the names of the segments in the current database.

- `You may not drop the log's last-chance threshold.`

  The threshold name and size you specified identify the last-chance threshold. You cannot drop this threshold.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropthreshold**.

Tables Used

sysobjects, syssegments, systhresholds

See Also

System procedures	**sp_addthreshold, sp_dboption, sp_helpthreshold, sp_thresholdaction**

sp_droptype

Function

Drops a user-defined datatype.

Syntax

`sp_droptype` ***`typename`***

Parameters

typename – is the name of a user-defined datatype that you own.

Examples

1. **`sp_droptype birthday`**

   Drops the user-defined datatype named *birthday*.

Comments

- Executing **sp_droptype** deletes a user-defined datatype from *systypes*.
- A user-defined datatype cannot be dropped if tables or other database objects reference it.

Messages

- `The type doesn’t exist or you don’t own it.`

  You do not own a user-defined datatype with that name.
- `Type is being used. You cannot drop it.`

  You cannot drop a user-defined datatype referenced by a table or other database object. Drop the tables and / or database objects first.
- `Type has been dropped.`

  The user-defined datatype no longer exists in the current database.

Permissions

Only the Database Owner or datatype owner can execute **sp_droptype**.

Tables Used

syscolumns, *sysobjects*, *systypes*, *sysusers*

See Also

System procedures	**sp_addtype, sp_rename**

sp_dropuser

Function

Drops a user from the current database.

Syntax

`sp_dropuser` ***`name_in_db`***

Parameters

name_in_db – is the user's name in the current database's *sysusers* table.

Examples

1. **`sp_dropuser albert`**

   Drops the user "albert" from the current database. "albert" can no longer use the database.

Comments

- **sp_dropuser** drops a user from the current database by deleting the user's row from *sysusers*.
- You cannot drop users who own objects in the database.
- You cannot drop users who have granted permissions to other users.
- You cannot drop the Database Owner from a database.
- If other users are aliased to the user being dropped, their aliases are also dropped. They no longer have access to the database.

Messages

- `The dependent aliases were also dropped.`

  Other users were aliased to the user being dropped. Their aliases have been dropped, and they can no longer access the database.
- `No user with the specified name exists in the current database.`

  The specified user doesn't exist in the current database.
- `User has been dropped from current database.`

  The specified user is no longer known to the database.

- `You cannot drop the 'database owner'.`

  The *name_in_db* is that of the Database Owner.

- `You cannot drop the 'guest' user from master or tempdb.`

  The "guest" user must exist in *master* and *tempdb* to allow the "guest" mechanism to work in other databases.

- `You cannot drop user because user '`*`name_in_db`*`' owns objects in database.`

  Users who own objects in the current database cannot be dropped. Drop the owned objects first. A list of datatypes and their owners is appears after this message.

- `You cannot drop user because user '`*`name_in_db`*`' owns thresholds in database.`

  Users who own thresholds in the current database cannot be dropped. Drop the owned thresholds first.

- `You cannot drop user because user '`*`name_in_db`*`' owns types in database.`

  Users who own user-defined datatypes in the current database cannot be dropped. Drop the owned datatypes first. A list of datatypes and their owners is appears after this message.

- `You cannot drop user because he or she owns grantable privileges and granted them to other users. Use sp_helprotect for more information.`

  Remove the grantable permissions from the user before he or she can be dropped.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropuser**.

Tables Used

#sysprotects1, #sysprotects2, master.dbo.spt_values, sysalternates, syscolumns, sysobjects, sysprotects, syssegments, systhresholds, systypes, sysusers

See Also

Commands	**grant, revoke, use**
System procedures	**sp_addalias, sp_adduser, sp_droplogin**

sp_estspace

Function

Estimates the amount of space required for a table and its indexes, and the time needed to create the index.

Syntax

```
sp_estspace table_name, no_of_rows [, fill_factor
   [, cols_to_max [, textbin_len [, iosec]]]]
```

Parameters

table_name – is the name of the table. It must already exist, and be in the current database.

no_of_rows – is the estimated number of rows that the table will contain.

fill_factor – is the index fillfactor. The default is null, which means the SQL Server uses its default fillfactor.

cols_to_max – is a comma-separated list of the variable length columns for which to use the maximum length instead of the average. The default is to use the average declared length of the variable-length columns.

textbin_len – is the length of all text and image columns, per row. The default value is 0. You only need to provide a value if the table stores text and / or image data. *text* and *image* columns are stored in a separate set of data pages from the rest of the table's data. The actual table row stores a pointer to the text or image value. **sp_estspace** provides a separate line of information about the size of the *text/image* pages for a row.

iosec – is the number of disk I / Os per second on this machine. The default is 30 I / Os per second.

Examples

1. **`sp_estspace titles, 10000, 50, "title,notes", 0, 25`**

```
name             type        idx_level Pages        Kbytes
---------------- ----------- --------- ------------ ------------
titles           data                0         3364         6728
titles           text/image          0            0            0
titleidind       clustered           0           21           43
titleidind       clustered           1            1            2
titleind         nonclustered        0         1001         2002
titleind         nonclustered        1           54          107
titleind         nonclustered        2            4            8
titleind         nonclustered        3            1            2

Total_Mbytes
-----------------
             8.68

name               type        total_pages  time_mins
------------------ ----------- ------------ ------------
titleidind         clustered           3386           13
titleind           nonclustered        1060            5
titles             data                   0            2
```

Calculates the space requirements for the *titles* table and its indexes, and the time required to create the indexes. The number of rows is 10,000, the fillfactor 50%, two variable-length columns are to be computed using the maximum size for the column, and the disk I/O speed is 25 I/Os per second.

2. **`declare @i int`**
   `select @i =  avg(datalength(pic)) from au_pix`
   `exec sp_estspace au_pix, 1000, null, null, @i`

```
au_pix has no indexes
 name            type        idx_level Pages     Kbytes
 --------------- ----------- --------- --------- ---------
 au_pix          data                0        31        63
 au_pix          text/image          0     21000     42000

 Total_Mbytes
 -----------------
            41.08
```

Uses the average length of existing *image* data in the *au_pix* table to calculate the size of the table with 1,000 rows. You can also provide this size as a constant.

3. **sp_estspace titles, 50000**

```
name            type         idx_level Pages        Kbytes
--------------- ------------ --------- ------------ ------------
titles          data                 0         4912         9824
titleidind      clustered            0           31           61
titleidind      clustered            1            1            2
titleind        nonclustered         0         1390         2780
titleind        nonclustered         1           42           84
titleind        nonclustered         2            2            4
titleind        nonclustered         3            1            2

Total_Mbytes
-----------------
            12.46

name               type         total_pages  time_mins
------------------ ------------ ------------ ------------
titleidind         clustered            4943           19
titleind           nonclustered         1435            8
```

Calculates the size of the *titles* table with 50,000 rows, using defaults for all other values.

Comments

- To estimate the amount of space required by a table and its indexes:

  1. Create the table.
  2. Create all indexes on the table.
  3. Run **sp_estspace**, giving the table name, the estimated number of rows that the table will contain, and the optional arguments, as needed.

  You do not need to insert data into the tables. **sp_estspace** uses information in the system tables—not the size of the data in the tables—to calculate the size of tables and indexes.

- If the **auto identity** option is turned on in a database, SQL Server automatically defines a 10-digit IDENTITY column in each new table that is created without specifying a **primary** key, a **unique** constraint, or an IDENTITY column. To estimate how much extra space is required by this column:

  1. In the master database, use **sp_dboption** to turn on the **auto identity** option for the database.
  2. Create the table.

3. Run **sp_estspace** on the table and record the results.
4. Drop the table.
5. Turn the **auto identity** option off for the database.
6. Recreate the table.
7. Rerun **sp_estspace** on the table and record the results.

- For information about tables or columns, use **sp_help** *tablename*.

Messages

- `Object does not exist in this database.`

  sp_estspace can be used only on tables that already exist in the current database.

- `Table contains text/image type columns. You must specify the total length per row for these columns in the argument list.`

  The table you specified contains *text* or *image* columns. Specify a length for these columns as the fifth argument. See Example 2.

Permissions

Any user can execute **sp_estspace**.

Tables Used

#col_table, #results, #times, syscolumns, sysindexes, sysobjects

See Also

Commands	**create index, create table**
System procedures	**sp_help**
Topics	Text and Image Datatypes

sp_extendsegment

Function

Extends the range of a segment to another database device, or extends an existing segment on the current database device.

Syntax

sp_extendsegment *segname, dbname, devname*

Parameters

segname – is the name of the existing segment, previously defined with **sp_addsegment**.

dbname – is the name of the database on which to extend the segment. *dbname* must be the name of the current database.

devname – is the name of the database device to add to the current database device range already included in *segname*.

Examples

1. **sp_extendsegment indexes, pubs2, dev2**

   This command extends the range of the segment *indexes* for the database *pubs2* on the database device *dev2*.

Comments

- After defining a segment, you may use it in the **create table** and **create index** commands to place the table or index on the segment. If you create a table or index on a particular segment, all the subsequent data for the table or index is located on the segment.
- To associate a segment with a database device, create or alter the database with a reference to that device. A database device can have more than one segment associated with it.
- A segment can be extended over several database devices.
- When you extend the *logsegment*, SQL Server recalculates its last-chance threshold.

Messages

- `Can’t run sp_extendsegment from within a transaction.`

  sp_extendsegment updates system tables, so it cannot be run from within a transaction.

- `Device '`*`devname`*`' is now exclusively used by '`*`segname`*`'.`
- `'`*`devname`*`' is reserved exclusively as a log device.`

  You can't create a segment on a database device that is dedicated to the database log.
- `No such device exists -- run sp_helpdb to list the devices for the current database.`

  The named device doesn't exist in *master.dbo.sysdevices*.
- `Segment extended.`

  sp_extendsegment succeeded. The segment named *segname* now includes space on the database device *devname*.
- `'`*`segname`*`' is not a valid identifier.`

  Segment names must conform to the rules for identifiers. They must begin with a letter, an underscore character (_), or a pound sign (#). After the first character, identifiers may include letters, underscores, pound signs, or dollar signs ($).
- `The specified device is not used by the database.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not used by the specified database. Segments may only be extended on database devices used by the database. Use **alter database** to extend a database on a device listed in the *master.dbo.sysdevices* table.
- `There is no such segment as '`*`segname`*`'.`

  The segment you have tried to extend does not exist. All segments for a database are listed in the *syssegments* table. Run **sp_helpsegment** to list them.
- `This command has been ignored. Extending the log segment on device '`*`devname`*`' would leave no space for creating objects in database '`*`database_name`*`'.`

  devname is the only, or the last, database device with space available for the database *database_name*.
- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use `*`database_name`*`' and try again.`

  sp_extendsegment can extend segments only in the database you are currently using. Issue the **use** command to open the database in which you want to extend a segment. Then run **sp_extendsegment** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_extendsegment**.

Tables Used

master.dbo.sysdatabases, sysdevices, master.dbo.sysusages, sysobjects, syssegments

See Also

Commands	**alter database, create index, create table**
System procedures	**sp_addsegment, sp_dropsegment, sp_helpdb, sp_helpdevice, sp_helpsegment, sp_placeobject**

sp_foreignkey

Function

Defines a foreign key on a table or view in the current database.

Syntax

```
sp_foreignkey tabname, pktabname, col1 [, col2] ...
   [, col8]
```

Parameters

tabname – is the name of the table or view that contains the foreign key.

pktabname – is the name of the table or view that has the primary key to which the foreign key applies. The primary key must already be defined.

col1 – is the name of the first column that makes up the foreign key. The foreign key must consist of at least one column, and can have a maximum of eight columns.

Examples

1. **`sp_foreignkey titles, publishers, pub_id`**

   The primary key of the *publishers* table is the *pub_id* column. The *titles* table also contains a *pub_id* column, which is a foreign key of *publishers*.

2. **`sp_foreignkey orders, parts, part, subpart`**

   The primary key of the *parts* table has been defined with **sp_primarykey** as the *partnumber* and *subpartnumber* columns. The *orders* table contains the columns *part* and *subpart*, which make up a foreign key of *parts*.

Comments

- **sp_foreignkey** adds the key to the *syskeys* table.
- Keys make explicit a logical relationship that is implicit in your database design.
- The number and order of columns that make up the foreign key must be the same as the number and order of columns that make up the primary key. The datatypes (and the lengths) of the

primary and foreign keys must agree, but the nulltypes need not agree.

- The installation process runs **sp_foreignkey** procedure on the appropriate columns of the system tables.
- To display a report on the keys that have been defined, execute **sp_helpkey**.

Messages

- `Datatypes of the column '`*`column_name`*`' in the keys are different.`

  The datatypes of the columns of the foreign key of *tabname* and the primary key of *pktabname* must be the same.
- `Foreign key table doesn't exist.`

  The table or view specified with the *tabname* parameter doesn't exist in the current database.
- `New foreign key added.`

  The foreign key has been defined and added to *syskeys*.
- `Only the owner of the table may define a foreign key.`

  You are not the owner of the table or view.
- `Primary key does not exist with the same number of columns as the foreign key.`

  The number of columns in the foreign key of *tabname* and in the primary key of *pktabname* must be the same.
- `Primary key table doesn't exist.`

  The table or view specified with the *pktabname* parameter doesn't exist in the current database, or doesn't have a primary key defined.
- `Table or view name must be in current database.`

  You cannot add foreign keys to a table or view in a different database.
- `The table does not have a column named '`*`column_name`*`'.`

  The table or view specified with the *tabname* parameter does not have a column of the specified name.
- `Primary key does not exist.`

  The primary key specified with the *col1*-*col8* parameters doesn't exist in the primary key table, or is not defined.

Permissions

You must be the owner of the table or view in order to define its foreign key.

Tables Used

#spconstrtab, syscolumns, sysindexes, syskeys, sysobjects, sysreferences

See Also

Commands	**create trigger**
System procedures	**sp_commonkey, sp_dropkey, sp_helpkey, sp_helpjoins, sp_primarykey**

sp_getmessage

Function

Retrieves stored message strings from *sysmessages* and *sysusermessages* for **print** and **raiserror** statements.

Syntax

sp_getmessage *message_num*, *@msg_var* output [, *language*]

Parameters

message_num – is the message number of the message to be retrieved.

@msg_var **output** – is the variable that receives the returned message text, followed by a space and the keyword **output**. The variable must have a datatype of *char, nchar, varchar,* or *nvarchar.*

language – is the language of the message to retrieve. Must be a valid language name in *syslanguages* table. If you include *language*, the message with the indicated *message_num* and *language* is retrieved. If not included, then the message for the default session language as indicated by the variable *@@langid* is retrieved.

Examples

1. **sp_getmessage 20001, @myvar output**

   Retrieves the message with the number 20001 from *sysusermessages*.

Comments

- Any appliacation can use **sp_getmessage**, and any user can read the messages stored in *sysmessages* and *sysusermessages*.

Messages

- `Message number must be greater than or equal to 17000.`

  You specified an invalid *message_num*.

- `'language' is not an official language name from syslanguages.`

  The *language* given is not a valid name in the *syslanguages* table.

- `Message number` *`message_number`* `does not exist in the` *`language`* `language.`

  The message number given does not exist in the *language* indicated.

Permissions

Any user can issue **sp_getmessage**.

Tables Used

master.dbo.syslanguages, *master.dbo.sysmessages*, *sysobjects*, *sysusermessages*

See Also

Commands	**print, raiserror**
System procedures	**sp_addmessage, sp_dropmessage**
Topics	Variables (Local and Global)

sp_help

Function

Reports information about a database object (any object listed in *sysobjects*), and about SQL Server-supplied or user-defined datatypes.

Syntax

sp_help [*objname*]

Parameters

objname – is the name of any object in *sysobjects* or of any user-defined datatype or system datatype in *systypes*. You cannot specify database names. *objname* can include tables, views, stored procedures, logs, rules, defaults, triggers, referential constraints, and check constraints. Use owner names if the object owner is not the user running the command and not the *dbo*.

Examples

1. **sp_help**

   Displays a brief listing of each object in *sysobjects*, giving the object's name, owner, and object type, and a brief listing of each user-defined datatype in *systypes*, giving the datatype's name, storage type, length, nulltype, default name, and rule name. Nulltype is 0 if null values are not permitted, 1 if null values are allowed.

2. **sp_help publishers**

```
Name                          Owner       Type
--------------------------    ----------- --------
publisher                     dbo         user table

Data_located_on_segment           When_created
------------------------------    --------------------
default                           Jan  1 1900 12:00AM

Column_name  Type    Length   Prec  Scale
----------   ------- ------  -----  -----
pub_id       char         4   NULL  NULL
pub_name     varchar     40   NULL  NULL
city         varchar     20   NULL  NULL
state        char         2   NULL  NULL
```

```
Nulls        Default_name   Rule_name     Identity
-----        ------------- ---------      --------
    0        NULL           NULL                 0
    1        NULL           NULL                 0
    1        NULL           NULL                 0
    1        NULL           NULL                 0

index_name       index_description                      index_keys
-------------    -------------------------------------  -------
pubind           clustered, unique located on default    pub_id

(1 row affected)
keytype    object      related_object object_keys
related_keys
-------    ----------  -------------- -----------------------------
--------------------------
primary    publishers  -- none --     pub_id, *, *, *, *, *, *, *
*, *, *, *, *, *, *, *
foreign    titles      publishers     pub_id, *, *, *, *, *, *, *
pub_id, *, *, *, *, *, *, *

(return status = 0)
```

Displays information about the *publishers* table.

3. **sp_help "mary.marytrig"**

```
Name         Owner             Type
------------ ----------------- ----------------
marytrig     mary              trigger

Data_located_on_segment When_created
----------------------- --------------------------
not applicable          Mar 20 1992  2:03PM
```

Displays information about the trigger *marytrig* owned by user *mary*. The quotes are needed, since "." is a special character.

4. **sp_help money**

```
Type_name  Storage_type Length Prec  Scale
---------- ------------ ------ ----- -----

money      money             8  NULL  NULL

Nulls   Default_name  Rule_name   Identity
-----   ------------  ---------   --------
    1   NULL          NULL               0

(return status = 0)
```

Displays information about the system datatype *money*.

5. **`sp_help identype`**

```
Type_name  Storage_type Length Nulls Default_name
--------- ------------ ------ ----- ----------
identype   numeric           4     0 NULL

Rule_name       Identity
---------       --------
NULL                   1

(return status = 0)
```

Displays information about the user-defined datatype *identype*. The report indicates the base type from which it was created, whether or not it allows nulls, the names of any rules and defaults bound to the datatype, and whether it has the IDENTITY property.

Comments

- **sp_help** looks for an object in the current database only.
- **sp_help** follows SQL Server's rule for finding objects:
  - If you don't specify an owner name, and you own an object with the specified name, **sp_help** uses that object.
  - If you don't specify an owner name, and don't own an object of that name, but the *dbo* does, **sp_help** uses the *dbo*'s object.
  - If neither you nor the *dbo* owns an object of that name, **sp_help** reports an error condition, even if an object exists in the database with that object name, but different owner.
  - If you and the *dbo* both own objects with the specified name, and you want to access the *dbo*'s object, specify the name in the format *dbo.objectname*.
  - Qulaify objects owned by database users other yourself and the *dbo* with the owner's name, as in Example 3.
- **sp_help** works on temporary tables if you issue it from *tempdb*.
- **sp_help** lists any indexes on a table, including indexes created by defining unique or primary key constraints in the **create table** or **alter table** statements. However, it does not describe any information about the integrity constraints defined for a table. Use **sp_helpconstraint** for information about any integrity constraints.

Messages

- `Object does not exist in this database.`

  The specified object does not exist in the current database.

- `Object must be in your current database.`

  sp_help only gives information about objects in the current database. Use **sp_helpdb** for information on the database itself.

Permissions

Any user can execute **sp_help**.

Tables Used

master.dbo.spt_values, syscolumns, sysindexes, sysmessages, sysobjects, systypes

See Also

System procedures	**sp_helpconstraint, sp_helpdb, sp_helpindex, sp_helpkey, sp_helprotect, sp_helpsegment, sp_helpuser**

sp_helpconstraint

Function

Reports information about any integrity constraints specified for a table. This information includes the constraint name and the definition of the default, unique / primary key constraint, referential constraint, or check constraint.

Syntax

sp_helpconstraint *tabname* [, detail]

Parameters

tabname – is the name of a table which has one or more integrity constraints defined by a **create table** or **alter table** statement.

detail – returns information about the constraint's user or error messages.

Examples

1. **sp_helpconstraint states**

```
name                      defn
------------------------  ----------------------------------------------------
states_popula_1088006907  CHECK (population > 1000000)
stateconstr               PRIMARY KEY INDEX (rank, abbrev): CLUSTERED,
                          FOREIGN REFERENCE
infoconstr                state_info FOREIGN KEY (rank, abbrev)
                          REFERENCES states (rank, abbrev)

(3 rows affected, return status = 0)
```

Displays the constraint information for the table *states*. *states* also has a foreign key to the table *state_info*. *states* and *state_info* are defined as follows:

```
create table states
(rank        smallint,
abbrev       char(2),
name         varchar(20) null,
population   int check (population > 1000000),
constraint stateconstr primary key (rank, abbrev))
```

```
create table state_info
(rank           smallint,
abbrev          char(2),
description     char(255),
comments        char(255) default "None",
constraint infoconstr foreign key (rank, abbrev)
    references states (rank, abbrev))
```

2. **sp_helpconstraint state_info, detail**

```
name                          type           defn            msg
----------------------------  -------------  --------------   -----
state_info_commen_1200007306   default value  DEFAULT "None"  NULL


infoconstr                           referential constraint
-----------------------------------  -----------------------------------
state_info FOREIGN KEY (rank, abbrev) REFERENCES states (rank, abbrev)
                                      standard system error message number: 547

(2 rows affected, return status = 0)
```

Displays more detailed information about the *state_info* constraints, including the constraint type and any constraint error messages.

Comments

- **sp_helpconstraint** prints the name and definition of the integrity constraint. The **detail** option returns information about the constraint's user or error messages.
- You can use **sp_helpconstraint** only for tables in the current database.
- **sp_helpconstraint** reports only the integrity constraint information about a table (defined by a **create table** or **alter table** statement). It does not report information about rules, triggers, or indexes created using the **create index** statement. Use **sp_help** to see information about rules, triggers, and indexes for a table.
- For constraints that do not have user defined messages, SQL Server reports the system error message associated with the constraint. Query *sysmessages* to obtain the actual text of that error message.

Messages

- `Object must be in current database.`

  sp_helpconstraint only provides information about objects in the current database. Use **sp_helpdb** for information about the database itself.

- `Object does not exist in this database.`

  The table you specified does not exist in the current database.

- `Object does not have any declarative constraints.`

  The specified object does not have any integrity constraints. Note that **sp_helpconstraint** only reports information about constraints defined by the **create table** or **alter table** statements.

Permissions

Any user can execute **sp_helpconstraint**.

Tables Used

syscolumns, syscomments, sysconstraints, sysindexes, sysobjects, sysreferences, sysusermessages

See Also

System procedures	**sp_help, sp_helpdb**

sp_helpdb

Function

Reports information about a particular database or about all databases.

Syntax

sp_helpdb [*dbname*]

Parameters

dbname – is the name of the database about whichto report information. Without this optional parameter **sp_helpdb** issues a report about all databases.

Examples

1. **sp_helpdb pubs2**

```
name    db_size owner  dbid  created          status
 -----  ------- -----  ----  ------------     ----------------------
 pubs2  2.0 MB  sa        4  Mar 05, 1993     aborttranwhenlogfull

 device_fragments          size      usage           free kbytes
------------------         ------    ------------  -------------
 master                    2.0 MB    data and log            576

 device                         segment
 ------------------------------ ------------------------------
 master                         default
 master                         logsegment
 master                         system
```

Displays information about the *pubs*2 database.

2. **sp_helpdb**

Displays information about all the databases on SQL Server, but does not include the *device_fragments* and *usage* columns.

Comments

- Executing **sp_helpdb** reports on the specified database when *dbname* is given, or on all databases in *master.dbo.sysdatabases* when no parameter is supplied.

- Executing **sp_helpdb** *dbname* from *dbname* reports segment information in addition to the information shown in the examples.

Messages

- `The specified database does not exist.`

  The specified database doesn't exist on SQL Server. Run **sp_helpdb** without the *dbname* parameter to see a list of all the databases.

Permissions

Any user can execute **sp_helpdb**.

Tables Used

master.dbo.spt_values, *sysdatabases*, *sysdevices*, *syslogins*, *sysmessages*, *syssegments*, *sysusages*, *#spdbdesc*

See Also

Commands	**alter database, create database**
System procedures	**sp_configure, sp_dboption, sp_renamedb**

sp_helpdevice

Function

Reports information about a particular device or about all SQL Server database devices and dump devices.

Syntax

```
sp_helpdevice [device_name]
```

Parameters

device_name – is the name of the device about which to report information. If you omit this parameter, information on all the devices appears.

Examples

1. **`sp_helpdevice`**

   Displays information about all the devices on SQL Server.

```
device_name    physical_name      description
------------   ------------------ ----------------------------------------
diskdump       null               disk, dump device
master         d_master           special, default disk, physical disk, 10 MB

               status   cntrltype    device_number     low      high
               ------   ----------   -------------     ----     -----
               16       2            0                 0        20000
               3        0            0                 0        5120
```

2. **`sp_helpdevice diskdump`**

   Reports information about the dump device named *diskdump*.

Comments

- Executing **sp_helpdevice** displays information on the specified device when *device_name* is given, or on all devices in *master.dbo.sysdevices* when no argument is given.
- The *sysdevices* table contains dump devices (tapes or disks to which you can dump databases with **dump database** and transaction logs with **dump transaction**) and database devices (devices on which databases are stored).

  Database devices can be designated default devices, which means that they can be used for database storage. This can occur when a user issues **create database** or **alter database** and does not specify a database device name, or gives the keyword **default**. To make a database device a default database device, execute the system procedure **sp_diskdefault**.

- Add database devices to the system with **disk init**.Add dump devices with **sp_addumpdevice**.
- The number in the *status* column corresponds to the status description in the *description* column.

  The *cntrltype* column specifies the controller number of the device. The *cntrltype* is 2 for disk or file dump devices, and 3 – 8 for tape dump devices. For database devices, it is almost always 0 (unless your installation has a special type of disk controller).

  The *device_number* column is 0 for dump devices, 0 for the master database device, and between 1 and 255 for other database devices.

  The *low* and *high* columns represent virtual page numbers, each of which is unique among all the devices on SQL Server.

Messages

- `No such i/o device exists.`

  The device name supplied for the *device_name* parameter doesn't exist on SQL Server. Run the procedure without the *device_name* parameter to see a list of all devices.

Permissions

Any user can execute **sp_helpdevice**.

Tables Used

master.dbo.spt_values, sysdevices, sysmessages, #spdevtab

See Also

Commands	**disk init, dump database, dump transaction, load database, load transaction**
System procedures	**sp_addumpdevice, sp_configure, sp_diskdefault, sp_dropdevice, sp_helpdb, sp_logdevice, sp_who**

sp_helpgroup

Function

Reports information about a particular group or about all groups in the current database.

Syntax

```
sp_helpgroup [grpname]
```

Parameters

grpname – is the name of a group in the database created with **sp_addgroup**.

Examples

1. **`sp_helpgroup hackers`**

   Displays information about the group "hackers":

```
Group_name  Group_id   Users_in_group  Userid
----------- ---------  --------------  ------
hackers     16384      ann             4
hackers     16384      judy            3
```

2. **`sp_helpgroup`**

   Displays information about all the groups in the current database:

```
Group_name      Group_id
--------------- --------
hackers         16384
public          0
```

Comments

- Executing **sp_helpgroup** reports on the specified group or, if no parameter is supplied, on all groups in the database.
- To get a report on the default group, "public," enclose the name "public" in single or double quotes ("public" is a reserved word).

Messages

- `No group with the specified name exists.`

  The specified group does not exist in the current database. Execute the procedure without the *grpname* parameter to see a list of all the groups in the database.

Permissions

Any user can execute **sp_helpgroup**.

Tables Used

syssrvroles, *sysusers*

See Also

Commands	**grant, revoke**
System procedures	**sp_addgroup, sp_changegroup, sp_dropgroup, sp_helprotect, sp_helpuser**

sp_helpindex

Function

Reports information about the indexes created on a table.

Syntax

sp_helpindex *tabname*

Parameters

tabname – is the name of a table in the current database.

Examples

1. **sp_helpindex sysobjects**

   Displays the types of indexes on the *sysobjects* table:

```
index_name     index_description                  index_keys
------------   ---------------------------------  ----------
titleidind     clustered, unique located on default title_id
titleind       nonclustered located on default     title

(2 rows affected, return status = 0)
```

Comments

- Executing **sp_helpindex** lists any indexes on a table, including indexes created by defining unique or primary key constraints of the **create table** or **alter table** statements.

Messages

- `Object does not exist in this database.`

  The name you gave for the *tabname* parameter does not exist in the current database.

- `Object does not have any indexes.`

  The table you named has no indexes.

- `Object must be in the current database.`

  The name you gave for the *tabname* parameter includes a database reference. Name references must be local to the current database.

Permissions

Any user can execute **sp_helpindex**.

Tables Used

master.dbo.spt_values, sysindexes, sysobjects, syssegments, #spindtab

See Also

Commands	**create index, drop index, update statistics**
System procedures	**sp_help, sp_helpkey**

sp_helpjoins

Function

Lists the columns in two tables or views that are likely join candidates.

Syntax

```
sp_helpjoins lefttab, righttab
```

Parameters

lefttab – is the first table or view.

righttab – is the second table or view. The order of the parameters does not matter.

Examples

1. **`sp_helpjoins sysobjects, syscolumns`**

   Displays a list of columns that are likely join candidates in the tables *sysobjects* and *syscolumns:*

```
a1              a2
        b1              b2
        c1              c2
        d1              d2
        e1              e2
        f1              f2
        g1              g2
        h1              h2
id              id
        NULL            NULL
        NULL            NULL
        NULL            NULL
        NULL            NULL
        NULL            NULL
        NULL            NULL
        NULL            NULL
```

Comments

- The column pairs that **sp_helpjoins** displays come from either of two sources. First, **sp_helpjoins** checks the *syskeys* table in the current database to see if any foreign keys have been defined on the two tables with **sp_foreignkey**, and then checks to see if any common keys have been defined on the two tables with **sp_commonkey**. If it doesn't find any foreign keys or common keys there, the procedure looks for any keys that may be reasonably joined: it checks for keys with the same user-defined datatypes, and if that fails, for columns with the same name and datatype.

- The **sp_helpjoins** procedure does not create any joins.

Messages

- `First table doesn't exist.`

  The table specified as the *lefttab* parameter is not a table or view in the current database.
- `Object must be in the current database.`

  lefttab and *righttab* must both be local to your current database.
- `Second table doesn't exist.`

  The table specified as the *righttab* parameter is not a table or view in the current database.

Permissions

Any user can issue **sp_helpjoins**.

Tables Used

#hjtab, syscolumns, syskeys, sysobjects

See Also

System procedures	**sp_commonkey, sp_foreignkey, sp_help, sp_helpkey, sp_primarykey**
Topics	Joins

sp_helpkey

Function

Reports information about a primary, foreign, or common key of a particular table or view, or about all keys in the current database.

Syntax

sp_helpkey [*objname*]

Parameters

objname – is the name of a table or view in the current database. If you don't specify a name, the procedure reports on all keys defined in the current database.

Examples

1. **sp_helpkey**

   Displays information on the keys defined in the current database:

```
keytype  object       related_object  object_keys          related_keys
-------  -------      --------------  -------------------  ------------------
primary  authors      -- none --      au_id,*,*,*,*,*,*,*  *,*,*,*,*,*,*,*
foreign  titleauthor  authors         au_id,*,*,*,*,*,*,*  au_id,*,*,*,*,*,*,*

(2 rows affected)
```

   The *object_keys* and *related_keys* columns refer to the names of the columns that make up the key.

Comments

- **sp_helpkey** lists information about all the primary, foreign, or common key definitions that reference the table *objname*, or about all the keys in the database if the parameter is omitted. Define these keys with the **sp_primarykey**, **sp_foreignkey**, and **sp_commonkey** system procedures.
- **sp_helpkey** does not provide information about the **unique** or **primary key** integrity constraints defined by a **create table** statement. Instead, use **sp_helpconstraint** to determine what constraints are defined for a table.
- Create keys to make explicit a logical relationship that is implicit in your database design. An application program can use the information.

- If you specify an object name, **sp_helpkey** follows SQL Server's rule for finding objects:
  - If you don't specify an owner name, and you own an object with the specified name, **sp_helpkey** uses that object.
  - If you don't specify an owner name, and you do not own an object of that name, but the *dbo* does, **sp_helpkey** uses the *dbo*'s object.
  - If neither you nor the *dbo* owns an object of that name, **sp_helpkey** reports an error condition, even if an object exists in the database with that object name, but different owner.
  - If you and the *dbo* both own objects with the specified name, and you want to access the *dbo*'s object, specify the name in the form *dbo.objectname*.
- Qualify objects owned by database users other than yourself and the *dbo* with the owner's name, as in "mary.myproc".

Messages

- `No defined keys for this object.`

  No primary, foreign, or common keys are defined for the specified table or view.
- `The name supplied for the` *`objname`* `parameter is not a table or view in the current database.`

  The table or view named doesn't exist in the current database.
- `Table or view name must be in current database.`

  The name supplied for the *objname* parameter included a database reference. Name references must be local to the current database.

Permissions

Any user can execute **sp_helpkey.**

Tables Used

master.dbo.spt_values, syskeys, sysobjects

See Also

Commands	**create trigger**
System procedures	**sp_commonkey, sp_foreignkey, sp_help, sp_primarykey**

sp_helplanguage

Function

Reports information about a particular alternate language or about all languages.

Syntax

sp_helplanguage [*language*]

Parameters

language – is the name of the alternate language that you want information about.

Examples

1. **sp_helplanguage french**

```
langid dateformat datefirst upgrade     name
      alias
      months
      shortmonths
      days
 ------ ---------- --------- ----------- -----------------------
     ----------------------------
     -----------------------------------------------------------
     -----------------------------------------------------------
     -----------------------------------------------------------
1 dmy              1          0 french
      french
      janvier,février,mars,avril,mai,juin,juillet,août,septembre,
          octobre,novembre,décembre
      jan,fév,mar,avr,mai,jui,juil,aoû,sep,oct,nov,déc
      lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
```

   This example displays information about the alternate language *french*.

2. **sp_helplanguage**

   This example displays information about all installed alternate languages.

Comments

- Executing **sp_helplanguage** reports on a specified language when *language* is given, or on all languages in *master.dbo.syslanguages* when no *language* is supplied.

Messages

- `*language* is not an official language name from syslanguages.`

  Use **sp_helplanguage** to see the list of official language names available on this SQL Server.

- `No alternate languages are available.`

  Use **sp_helplanguage** to see the list of official language names available on this SQL Server.

- `us_english is always available, even though it is not in master.dbo.syslanguages.`

  This message appears at the end of each report from **sp_helplanguage.**

Permissions

Any user can execute **sp_helplanguage**.

Tables Used

master.dbo.syslanguages, *sysobjects*

See Also

System procedures	**sp_addlanguage, sp_droplanguage, sp_setlangalias**

sp_helplog

Function

Reports the name of the device that contains the first page of the log.

Syntax

```
sp_helplog
```

Parameters

None.

Examples

1. **`sp_helplog`**

   ```
   In database 'master', the log starts on device
   'master'.
   ```

Comments

- **sp_helplog** displays the name of the device that contains the first page of the log in the current database.

Messages

- `In database '`*`database_name`*`', the log starts on device '`*`device_name`*`'.`

  The named device contains the first page of the database's log.

Permissions

Any user can execute **sp_helplog**.

Tables Used

master.dbo.sysdevices, master.dbo.sysusages, sysindexes, sysobjects

See Also

Commands	**alter database, create database**
System procedures	**sp_helpdevice, sp_logdevice**

sp_helpremotelogin

Function

Reports information about a particular remote server's logins or about all remote servers' logins.

Syntax

sp_helpremotelogin [*remoteserver* [,*remotename*]]

Parameters

remoteserver – is the name of the server about which to report remote login information.

remotename – is the name of a particular remote user on the remote server.

Examples

1. **sp_helpremotelogin GATEWAY**

   Displays information about all the remote users of the remote server GATEWAY.

2. **sp_helpremotelogin**

   Displays information about all the remote users of all the remote servers known to the local server.

Comments

- **sp_helpremoteserver** reports on the remote logins for the specified server when *remoteserver* is given, or on all servers when no parameter is supplied.

Messages

- `There are no remote logins.`
- `There are no remote logins defined.`

  There are no remote logins for any remote server in *master.dbo.sysremotelogins*.
- `There are no remote logins for '`*`remotename`*`'.`

  The remote server has no entries in the *master.dbo.sysremotelogins* table.

- `There are no remote logins for '`*`remotename`*`' on remote server '`*`remoteserver`*`'.`

  There is no remote login for the user *remoteuser* on the remote server *remoteserver*.

- `There are no remote logins for the remote server '`*`remoteserver`*`'.`

  The specified server isn't listed in *master.dbo.sysservers*. Run the procedure without the *remoteserver* parameter to see remote login information for all the servers. To get a list of all the servers, run **sp_helpserver**.

- `There are no remote servers defined.`

  The *master.dbo.sysservers* table has no entries for remote servers.

Permissions

Any user can execute **sp_helpremotelogin**.

Tables Used

master.dbo.spt_values, master.dbo.sysmessages, master.dbo.sysremotelogins, master.dbo.sysservers, sysobjects

See Also

System procedures	**sp_addremotelogin, sp_dropremotelogin, sp_helpserver**

sp_helprotect

Function

Reports on permissions for database objects, users, or groups.

Syntax

sp_helprotect [*name* [, *name_in_db* [, "grant"]]]

Parameters

name – is either the name of the table, view, or stored procedure, or the name of a user or group in the current database. If you do not provide a name, **sp_helprotect** reports on all permissions in the database.

name_in_db – is a user's name in the current database.

grant – displays the privileges granted to *name* **with grant option**.

Examples

1.
   ```
   grant select on titles to judy
   grant update on titles to judy
   revoke update on titles(price) from judy
   grant select on publishers to judy
   with grant option
   ```

   After this series of **grant** and **revoke** statements, executing **sp_helprotect titles** results in this display:

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   titles     All         FALSE
dbo     judy   Grant   Update   titles     advance     FALSE
dbo     judy   Grant   Update   titles     notes       FALSE
dbo     judy   Grant   Update   titles     pub_id      FALSE
dbo     judy   Grant   Update   titles     pubdate     FALSE
dbo     judy   Grant   Update   titles     title       FALSE
dbo     judy   Grant   Update   titles     title_id    FALSE
dbo     judy   Grant   Update   titles     total_sales FALSE
dbo     judy   Grant   Update   titles     type        FALSE
dbo     judy   Grant   Select   publishers all         TRUE
```

2. **`grant select, update on titles(price, advance)`**
   `to mary`
   `with grant option`

   After this **grant** statement, **sp_helprotect titles** displays the following:

```
grantor    grantee    type    action  object  column  grantable
-------    -------    ------  ------- ------  ------  ---------
dbo        mary       Grant   Select  titles  advance  TRUE
dbo        mary       Grant   Select  titles  price    TRUE
dbo        mary       Grant   Update  titles  advance  TRUE
dbo        mary       Grant   Update  titles  price    TRUE
```

3. **`sp_helprotect judy`**

   Displays all the permissions that "judy" has in the database.

Comments

- **sp_helprotect** reports permissions on a database object. If you supply the *name_in_db* parameter, only those user's permissions on the database object are reported. If *name* is not an object, **sp_helprotect** checks to see if it is a user or group, and if it is, **sp_helprotect** lists the permissions for the user or group.
- **sp_helprotect** looks for objects and users in the current database only.

Messages

- `Object must be in current database.`

  The name supplied for the *name* parameter included a reference to a database. The name must be local to the database.
- `No user with the specified name exists in the current database.`

  The name supplied for *name_in_db* is not a user or group in the current database.
- `No such object or user exists in the database.`

  The name supplied for the *name* parameter is not an object, user, or group in the current database.

Permissions

Any user can execute **sp_helprotect**.

Tables Used

#sysprotects1, #sysprotects2, master.dbo.spt_values, syscolumns, sysobjects, sysprotects, sysusers

See Also

Commands	**grant, revoke**
System procedures	**sp_help**

sp_helpsegment

Function

Reports information on a particular segment or on all of the segments in the current database.

Syntax

```
sp_helpsegment [segname]
```

Parameters

segname – is the name of the segment about which you want information. If you omit this parameter, information on all the segments in the current database appears.

Examples

1. **`sp_helpsegment segment3`**

   Reports information about the segment named *segment3*, including which database tables and indexes use that segment.

2. **`sp_helpsegment "default"`**

   Reports information about the *default* segment. Notice that the keyword default must be enclosed in quotes.

3. **`sp_helpsegment logsegment`**

   Reports information about the segment on which the transaction log is stored.

Comments

- **sp_helpsegment** displays information on the specified segment when *segname* is given, or on all segments in the current database when no argument is given.
- Add segments to the *syssegments* table in the current database with **sp_addsegment**.

Messages

- `There is no such segment as segname.`

  The segment name supplied for the *segname* parameter doesn't exist in the *syssegments* table. Run **sp_helpsegment** without the *segname* parameter to see a list of all segments for the current database.

Permissions

Any user can execute **sp_helpsegment**.

Tables Used

master.dbo.sysdevices, *master.dbo.sysusages*, *sysindexes*, *sysobjects*, *syssegments*

See Also

System procedures	**sp_addsegment, sp_dropsegment, sp_extendsegment, sp_helpdb, sp_helpdevice**

sp_helpserver

Function

Reports information about a particular remote server or about all remote servers.

Syntax

sp_helpserver [*server*]

Parameters

server – is the name of the remote server that you want information about.

Examples

1. **sp_helpserver GATEWAY**

   Displays information about the remote server GATEWAY.

2. **sp_helpserver**

   Displays information about all the remote servers known to the local server.

Comments

- **sp_helpserver** reports on all servers in *master.dbo.sysservers*, or on a specified remote server when *server* is given.

Messages

- `There are no remote servers defined.`

  This SQL Server has no remote servers defined.

- `There is not a server named` *`server`*`.`

  The specified server isn't listed in *master.dbo.sysservers*. Run the procedure without the *server* parameter to see a list of all the servers.

Permissions

Any user can execute **sp_helpserver.**

Tables Used

#spt_server, master.dbo.spt_values, master.dbo.sysservers, sysobjects

See Also

System procedures	**sp_addserver, sp_dropserver, sp_helpremotelogin, sp_serveroption**

sp_helpsort

Function

Displays SQL Server's default sort order and character set.

Syntax

`sp_helpsort`

Parameters

None

Examples

1. **`sp_helpsort`**

   For Class 1 (single byte) character sets, **sp_helpsort** displays the name of the server's default sort order, its character set, and a table of its primary sort values. On a 7-bit terminal it appears as follows:

```
Sort Order Description
------------------------------------------------------------------
 Character Set = 1, iso_1
     ISO 8859-1 (Latin-1) - Western European 8-bit character set.
 Sort Order = 50, bin_iso_1
     Binary sort order for the ISO 8859/1 character set (iso_1).
Characters, in Order

------------------------------------------------------------------
   ! “ # $ % & ‘ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
   @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
   ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
   ! “ # $ % & ‘ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
   @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
   ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

On an 8-bit terminal it appears as follows:

```
Sort Order Description
-----------------------------------------------------------------
 Character Set = 1, iso_1
     ISO 8859-1 (Latin-1) - Western European 8-bit character set.
 Sort Order = 50, bin_iso_1
     Binary sort order for the ISO 8859/1 character set (iso_1).
Characters, in Order

-----------------------------------------------------------------
   ! “ # $ % & ‘ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
  @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
  ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
  ¡ ¢ £ ¤ ¥ | § ¨ © ª « ¬ – ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À
  Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï D Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Y P ß à
  á â ã ä å æ ç è é ê ë ì í î ï ∂ ñ ò ó ô õ ö ÷ ø ù ú û ü y p ÿ
```

For a Class 2 (multibyte) character set, the characters are not listed, but a description of the character set is included. For example:

```
Sort Order Description
-----------------------------------------------------------------
Character Set = 140, euc_jis
    Japanese. Extended Unix Code mapping for JIS-X0201
    (hankaku katakana) and JIS-X0208 (double byte) roman,
     kana, and kanji.
    Class 2 character set
Sort Order = 50, bin_eucjis
    Binary sort order for Japanese using the EUC JIS
      character set as a basis.
```

Comments

- Binary sort order is the default.

Messages

- `Unknown character set:` *`character_set`*

  sp_helpsort does not recognize *character_set*, but displays the characters in order.

Permissions

Any user can execute **sp_helpsort.**

Tables Used

#helpsort1, *#helpsort2*, *#helpsort3*, *master.dbo.syscharsets*, *master.dbo.syscurconfigs*, *sysobjects*

sp_helptext

Function

Prints the text of a system procedure, trigger, view, default, rule, or integrity check constraint.

Syntax

sp_helptext *objname*

Parameters

objname – is the name of the object for which to display the **create** text. It must be in the current database.

Examples

1. **sp_helptext pub_idrule**

   Displays the text of *pub_idrule*. Since this rule is in the *pubs2* database, this command must be issued from *pubs2*.

```
# Lines of Text
---------------
1

(1 row affected)

text

-------------------------------------------------------------
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622", "1756")
or @pub_id like "99[0-9][0-9]"

(1 row affected, return status = 0)
```

2. **sp_helptext sp_helptext**

   Displays the text of **sp_helptext**. Since system procedures are stored in *sybsystemprocs*, execute this command from *sybsystemprocs*.

Comments

- **sp_helptext** prints out the number of rows in *syscomments* (255 characters long each) that the object occupies, followed by the **create** text of the object.

- **sp_helptext** only looks for the text in the *syscomments* table of the current database.

Messages

- `Object must be in the current database.`

  The *objname* parameter included a database name reference. The *objname* must be in the current database.
- `There is no text for object` *`object_name`*`.`

  objname is an object in the current database that does not have text in *syscomments* (a table or index, for example).

Permissions

Any user can execute **sp_helptext**.

Tables Used

syscomments, *sysobjects*

See Also

Commands	**create default, create procedure, create rule, create trigger, create view**
System procedures	**sp_help**

sp_helpthreshold

Function

Reports the segment, free-space value, status, and stored procedure associated with all thresholds in the current database or all thresholds for a particular segment.

Syntax

sp_helpthreshold [*segment_name*]

Parameters

segment_name – is the name of a segment in the current database.

Examples

1. **`sp_helpthreshold logsegment`**

   Shows all thresholds on the log segment.

2. **`sp_helpthreshold`**

   Shows all thresholds on all of the segments of the current database.

3. **`sp_helpthreshold "default"`**

   Shows all thresholds on the default segment. Note the use of quotes around the reserved word “default”.

Comments

- **sp_helpthreshold** displays threshold information for all the segments in the current database. If you provide the name of a segment, **sp_helpthreshold** lists all of the thresholds on that segment.
- The *status* column is 1 for the last-chance threshold and 0 for all other thresholds. Databases that do not store their transaction logs on a separate segment have no last-chance threshold.

Messages

- `Database '`*`dbname`*`' has no thresholds--table '`*`systhresholds`*`' does not exist.`

  The *systhresholds* table is missing. This table is created when the database is created (or an upgrade to Release 10 is performed), and must not be removed.

- `Segment '`*`segment_name`*`' does not exist.`

  Use **sp_helpsegment** to see the names of segments in a database.

Permissions

Any user can execute **sp_helpthreshold**.

Tables Used

sysobjects, *syssegments*, *systhresholds*

See Also

System procedures	**sp_addthreshold, sp_dropthreshold, sp_helpsegment, sp_modifythreshold, sp_thresholdaction**

sp_helpuser

Function

Reports information about a particular user or about all users in the current database.

Syntax

sp_helpuser [*name_in_db*]

Parameters

name_in_db – is the user's name in the current database.

Examples

1. **sp_helpuser**

   Displays information about all users in the current database:

```
Users_name ID_in_db    Group_name  Login_name  Default_db
---------  --------    ----------  ----------  ----------
ann        4           hackers     ann         master
dbo        1           public      sa          master
guest      2           public      NULL        NULL
judy       3           hackers     judy        master
```

2. **sp_helpuser dbo**

   Displays information about the Database Owner (user name "dbo"):

```
Users_name     ID_in_db    Group_name  Login_name  Default_db
----------     --------    ----------  ----------  ----------
dbo            1           public      sa          master

Users aliased to user.
Login_name
------------------------------
andy
christa
howard
linda
```

Comments

- **sp_helpuser** reports information on all users of the current database. If you specify a *name_in_db*, **sp_helpuser** reports information only on the specified user.

- If the specified user is not listed in the current database's *sysusers* table, **sp_helpuser** checks to see if the user is aliased to another user or is a group name.

Messages

- `The name supplied is a group name.`

  The name specified for the *name_in_db* parameter is a group name.

- `The name supplied is aliased to another user.`

  The name supplied is not a user in the database, but is aliased to a user in the database.

- `The name supplied is not a user, group, or aliased.`

  The name supplied is unknown in the database as a login, user, or group.

- `Users aliased to user.`

  If the user has other users aliased to him or her, the names of the other users are listed. (See Example 2.)

Permissions

Any user can execute **sp_helpuser**.

Tables Used

master.dbo.syslogins, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**grant, revoke, use**
System procedures	**sp_adduser, sp_dropuser, sp_help, sp_helpgroup**

sp_indsuspect

Function

Checks user tables for indexes marked as suspect during recovery following a sort order change.

Syntax

sp_indsuspect [*table_name*]

Parameters

table_name – is the name of the user table to check.

Examples

1. **sp_indsuspect newaccts**

   Checks the table *newaccts* for indexes marked as suspect.

Comments

- **sp_indsuspect** with no parameter creates a list of all tables in the current database with indexes that need to be rebuilt as a result of a sort order change. With a *table_name* parameter, **sp_indsuspect** checks the indicated user table for indexes marked as suspect during recovery following a sort order change.
- Use **sp_indsuspect** to list all suspect indexes. The table owner or a System Administrator can use **dbcc reindex** to check the integrity of the listed indexes and to rebuild them if necessary.

Messages

- `Suspect indexes in database` *`database_name`*`:`
  `Own.Tab.Ind (Obj_ID, Ind_ID) =`
  `owner``.`*`table`*`.`*`index_name`*`(`*`object_id`*`, `*`index_id`*`)`

  The listed indexes are suspect and should be reindexed using **dbcc reindex**.
- `There are no suspect indexes in database` *`database_name`*`.`

  No tables in the current database contain suspect indexes.
- `Table must be in the current database.`

  sp_indsuspect only checks the current database for suspect indexes. You cannot use a fully qualified table name to check tables in another database. To check for suspect indexes in another database, use the **use** command to access the database.

- `There is no table named` *`table_name`* `in the current database.`

  The current database does not contain the table name you specified. Check the table name and rerun **sp_indsuspect**.

- `Suspect indexes on table` *`table_name`*`:`
  `Own.Tab.Ind (Obj_ID, Ind_ID) =`
  `owner.table.index_name(object_id, index_id)`

  The listed indexes are suspect and should be reindexed using **dbcc reindex**.

- `There are no suspect indexes on table` *`table_name`*`.`

  The specified table does not contain suspect indexes.

Permissions

Any user can execute **sp_indsuspect**.

Tables Used

sysindexes, *sysobjects*, *sysusers*

See Also

Commands	**dbcc**

sp_lock

Function

Reports information about processes that currently hold locks.

Syntax

sp_lock [*spid1* [, *spid2*]]

Parameters

spid1 – is the SQL Server process ID number from the *master.dbo.sysprocesses* table. Run **sp_who** to get the *spid* of the lock.

spid2 – is another SQL Server process ID number to check for locks.

Examples

1. sp_lock

```
The class column will display the cursor name for locks
associated with a cursor for the current user and the cursor id
for other users.
spid  locktype      table_id    page    dbname  class
----  ------------  ----------  ----    ------  ---------------
1     Ex_intent     1308531695  0       master  Non cursor lock
1     Ex_page       1308531695  761     master  Non cursor lock
1     Sh_intent     380528389   0       master  Cursor Id 327681
1     Update_page   380528389   3752    master  Cursor Id 327681
5     Ex_intent     144003544   0       userdb  Non cursor lock
5     Ex_page       144003544   509     userdb  Non cursor lock
5     Ex_page       144003544   1419    userdb  Non cursor lock
5     Ex_page       144003544   1420    userdb  Non cursor lock
5     Ex_page       144003544   1440    userdb  Non cursor lock
5     Sh_page       144003544   1440    userdb  Non cursor lock
5     Sh_table      144003544   0       userdb  Non cursor lock
5     Update_page   144003544   1440    userdb  Non cursor lock
5     Sh_intent     380528389   0       master  objects_crsr
4     Ex_table      240003886   0       pubs2   Non cursor lock
4     Sh_intent     112003436   0       pubs2   Non cursor lock
4     Ex_intent-blk 112003436   0       pubs2   Non cursor lock
```

Displays information on all locks currently held in SQL Server.

2. sp_lock 1

```
The class column will dispaly the cursor name for locks
associated with a cursor for the current user and the cursor id
for other users.
spid  locktype      table_id    page    dbname  class
----  ------------  ----------  ----    ------  ---------------
1     Ex_intent     1308531695  0       master  Non cursor lock
1     Ex_page       1308531695  761     master  Non cursor lock
```

Displays information on locks currently held on *spid1*.

Comments

- **sp_lock** with no parameters reports information on all processes that currently hold locks.
- The only user control over locking is through the use of the **holdlock** keyword in the **select** statement.
- Use the **object_name()** function to derive a table's name from its ID number.
- The *locktype* column indicates not only whether the lock is a shared lock ("Sh" prefix), an exclusive lock ("Ex" prefix) or an update lock, but also whether it is held on a table ("table" or "intent") or on a page.

  The "blk" suffix in the *locktype* column indicates that this process is blocking another process which needs to acquire a lock. As soon as this process completes, the other process(es) move forward. The "demand" suffix indicates when the process is attmepting to acquire an exclusive lock.
- In general, read operations acquire **shared** locks, while write operations acquire **exclusive** locks. **Update** locks are created at the page level. Update locks are acquired during the initial portion of an update operation when the pages are being read. The update locks are compatible with shared locks. Later, if the pages are changed, the update locks are promoted to exclusive locks.

  An **intent** lock indicates the intention to acquire a shared or exclusive lock on a data page. An intent lock prevents another transaction from acquiring an exclusive lock on the table that contains that page.

  A **demand** lock prevents any more shared locks from being set. It indicates that a transaction is next in line to lock a table or page. Demand locks are necessary because shared locks can overlap, so that read transactions keep monopolizing a table or page, forcing a write transaction to wait indefinitely. After waiting on four different read transactions, a write transaction is given a demand lock. As soon as the existing read transactions finish, the write transaction is allowed to proceed. Any new read transactions then have to wait for the write transaction to finish.

- The *class* column indicates whether a lock is associated with a cursor. It displays one of the following:
  - "Non Cursor Lock" indicates the lock is not associated with a cursor.
  - "Cursor Id *number*" indicates the lock is associated with cursor ID *number* for that SQL Server process ID.
  - "*cursor_name*" indicates the lock is associated with cursor *cursor_name* owned by the current user executing **sp_lock**.

Messages

- `The class column will display the cursor name for locks associated with a cursor for the current user and the cursor id for other users.`

Permissions

Any user can execute **sp_lock**.

Tables Used

master.dbo.spt_values, master.dbo.syslocks, sysobjects

See Also

Commands	**kill, select** (**holdlock** keyword)
System procedures	**sp_who**

sp_locklogin

Function

Locks a SQL Server account so that the user cannot log in, or displays a list of all locked accounts.

Syntax

sp_locklogin [*login_name*, "{lock | unlock}"]

Parameters

login_name – is the name of the account to lock or unlock.

lock | **unlock** – specifies whether to lock or unlock *login_name*.

Examples

1. **sp_locklogin charles, "lock"**

   Locks the login account for the user “charles.”

2. **sp_locklogin**

   Displays a list of all locked accounts.

Comments

- **sp_locklogin** with no parameters returns a list of all the locked accounts.
- *login_name* must be the name of an existing valid account.
- You can lock an account that is currently logged in. The user receives a warning that his or her account has been locked, but is not locked out of the account until he or she logs out.
- A locked account can be specified as a Database Owner and can own objects in any database.
- Locking an account that is already locked or unlocking an unlocked account has no effect.
- When locking a System Security Officer’s login account, **sp_locklogin** verifies that at least one other unlocked System Security Officer’s account exists. Similarly, **sp_locklogin** verifies that there is always an unlocked System Administrator’s account. An attempt to lock the last remaining unlocked System Administrator or System Security Officer account causes **sp_locklogin** to return an error message and fail.

Messages

- `Can’t run sp_locklogin from within a transaction.`

  sp_locklogin modifies system tables, so it cannot be run from within a transaction.

- `No such account -- nothing changed.`

  You have specified an invalid *login_name*.

- `Locked account(s):`

  Lists all locked accounts.

- `Account unlocked.`

  You have successfully unlocked the account.

- `Account locked.`

  You have successfully locked the account.

- `Warning: the specified account is currently active.`

  The account you’ve specified is currently logged in; it will be locked when the user next tries to log in.

- `Cannot lock the last remaining unlocked SSO|SA account.`

  An active System Administrator and System Security Officer account must always exist.

Permissions

System Administrators and System Security Officers can use **sp_locklogin** to lock logins and to display locked logins.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, sysobjects

See Also

System procedures	**sp_addlogin, sp_modifylogin, sp_password**
Topics	Login Management

sp_logdevice

Function

Puts the system table *syslogs*, which contains the transaction log, on a separate database device.

Syntax

```
sp_logdevice dbname, device_name
```

Parameters

dbname – is the name of the database whose *syslogs* table you want to put on a specific logical device.

device_name – is the logical name of the device on which you want to put the *syslogs* table. This device must be a database device associated with the database (named in **create database** or **alter database**). Run **sp_helpdb** for a report on the database's devices.

Examples

1.
   ```
   create database products on default = 10, logs = 2
   sp_logdevice products, logs
   ```

   Creates the database *products* and puts the table *products.syslogs* on the database device *logs*.

Comments

➤ ***Note***

The **sp_logdevice** procedure affects only future allocations of space for *syslogs*. This creates a window of vulnerability during which the first pages of your log remain on the same device as your data. Therefore, the preferred method of placing a transaction log on a separate device is with the **log on** option to **create database**, which immediately places the entire transaction log on a separate device.

- Place transaction logs on separate database devices, for both recovery and performance reasons.

  A very small, non-critical database could keep its log together with the rest of the database. Such databases use **dump database** to backup the database and log and **dump transaction with truncate_only** to truncate the log.

- **dbcc checkalloc** and **sp_helplog** show some pages for *syslogs* still allocated on the database device until after the next **dump transaction**. After that, the transaction log is completely transferred to the device named when you executed **sp_logdevice**.
- The size of the device required for the transaction log varies according to the amount of update activity and the frequency of transaction log dumps. As a rule of thumb, allocate to the log device 10% to 25% of the space you allocate to the database itself.
- To increase the amount of storage allocated to the transaction log use **alter database**. If you used the **log on** option to **create database** to place a transaction log on a separate device, use:

  ```
  sp_extendsegment segname, device_name
  ```

  to increase the size of the log segment. If you did not use **log on**, execute **sp_logdevice**.

  The device or segment on which you put *syslogs* is used **only** for the *syslogs* table. If you want to increase the amount of storage space allocated for the rest of the database, specify any device other than the log device when you issue the **alter database** command.
- Use the **disk init** command to format a new database device for databases or transaction logs.
- See the *System Administration Guide* for details.

Messages

- `No such database -- run sp_helpdb to list databases.`

  No database with the supplied name exists. Run **sp_helpdb** to get a list of databases.
- `No such device exists -- run sp_helpdevice to list the SQL Server devices.`

  The *device_name* device doesn't exist on SQL Server.
- `syslogs moved.`

  The procedure was successful and the *syslogs* table is now located on the *device_name* device.
- `The last-chance threshold for database` *`dbname`* `is now` *`n`* `pages.`

  SQL Server created a last-chance threshold for the log segment of the database. When the amount of free space on the log segment falls below *n* pages, SQL Server automatically executes

sp_thresholdaction. Use **sp_modifythreshold** to change the procedure associated with the last-chance threshold.

- `Could not update the last-chance threshold for database` *`dbname`*`.`

  SQL Server was unsuccessful in creating a last-chance threshold for the log segment. Your *systhresholds* table may be corrupt.

- `The specified device is not used by the database.`

  The database *dbname* has no space allocated on the device *device_name*.

- `This command has been ignored. The device specified is the only non-log device available for the database and cannot be made log-only.`

  The *device_name* you specified is the only, or the last, database device with space available for *dbname*. Making it a log device would leave no space for creating any more objects in the database.

Permissions

Only the Database Owner or System Administrator can execute **sp_logdevice**.

Tables Used

master.dbo.sysdatabases, *master.dbo.sysdevices*, *master.dbo.sysusages*, *sysobjects*

See Also

Commands	**alter database, create database, dbcc, disk init, dump database, dump transaction, select**
System procedures	**sp_extendsegment, sp_helpdevice**

sp_modifylogin

Function

Modifies the default database, default language, or full name for a SQL Server login account.

Syntax

```
sp_modifylogin login_name, option, value
```

Parameters

login_name – is the login account to modify.

option – specifies the name of the option to change. The options are:

Option	Definition
defdb	The “home” database to which the user is connected when he or she logs in.
deflanguage	The official name of the user’s default language.
fullname	The user’s full name.

Table 1-13: Options for sp_modifylogin

value – is the new value for the specified option.

Examples

1. `sp_modifylogin sarah, defdb, "pubs2"`

   Changes the default database for “sarah” to *pubs2*.

2. `sp_modifylogin claire, deflanguage, "french"`

   Sets the default language for “claire” to French.

3. `sp_modifylogin clemens, fullname, "Samuel Clemens"`

   Changes user “clemens” full name to “Samuel Clemens.”

Comments

- Set a default database, language, or full name either with **sp_modifylogin** or with **sp_addlogin** when first adding the user’s login to SQL Server.
  - If you do not specify a default database, the user’s default is *master*.
  - If you do not specify a language, the user’s default language is set to the server’s default language.

- If you do not specify a full name, that column in *syslogins* remains blank.

- After **sp_modifylogin** is executed, the user is connected to the new *defdb* the next time he or she logs in. The user, however, cannot access the database until the Database Owner gives the user access through **sp_adduser** or **sp_addalias**, or if there is a “guest” user in the database’s *sysusers* table. If the user does not have access to the database by any of these means, she or he is connected to *master* and an error message appears.
- If a user’s default database is dropped, or if the user is dropped from the database, the user is connected to *master* on his or her next login, and an error message appears.
- If a user’s default language is dropped from the server, the server-wide default language is used as the initial language setting, and a message appears.

Messages

- `Can’t run sp_modifylogin from within a transaction.`

  sp_modifylogin modifies system tables so it cannot be run from within a transaction.
- `No such account -- nothing changed.`

  You specified a nonexistent account name.
- `Column changed.`

  sp_modifylogin executed successfully.
- `Column name invalid -- nothing changed.`

  You specified an invalid name for the *option* parameter.

Permissions

Only a System Administrator can execute **sp_modifylogin.**

Tables Used

master..syslogins, sysobjects

See Also

System procedures	**sp_addlogin, sp_password**
Topics	Login Management

sp_modifythreshold

Function

Modifies a threshold by associating it with a different threshold procedure, level of free space, or segment. You **cannot** use **sp_modifythreshold** to change the amount of free space or the segment name for the last-chance threshold.

Syntax

```
sp_modifythreshold database, segment, free_pages
   [, new_procedure] [, new_free_pages]
   [, new_segment]
```

Parameters

database – is the database for which to change the threshold. This must be the name of the current database.

segment – is the segment for which to monitor free space. Use quotes when specifying the "default" segment.

free_pages – is the number of free pages at which the threshold is crossed. When free space in the segment falls below this level, SQL Server executes the associated stored procedure.

new_procedure – is the new stored procedure to execute when the threshold is crossed. The procedure can be located in any database on the current SQL Server or on an Open Server. Thresholds cannot execute procedures on remote SQL Servers.

new_free_pages – is the new number of free pages to associate with the thrshold. When free space in the segment falls below this level, SQL Server executes the associated stored procedure.

new_segment – is the new segment for which to monitor free space. Use quotes when specifying the "default" segment.

Examples

1. **sp_modifythreshold mydb, "default", 200, NULL, 175**

   Modifies a threshold on the *default* segment of the *mydb* database. The threshold procedure now executes when free space on the segment falls below 175 pages, rather than 200 pages. The NULL is a place holder indicating that the procedure name is not being changed.

2. **`sp_modifythreshold mydb, data_seg, 250, new_proc`**

   Modifies a threshold on the *data_seg* segment of *mydb* so that it executes the *new_proc* procedure.

Comments

- See the *System Administration Guide* for more information about using thresholds.

Crossing a Threshold

- When a threshold is crossed, SQL Server executes the associated stored procedure. SQL Server uses the following search path for the threshold procedure:
  - If the procedure name does not specify a database, SQL Server looks in the database in which the threshold was crossed.
  - If the procedure is not found in this database and the procedure name begins with *sp_*, SQL Server looks in the *sybsystemprocs* database.

  If the procedure is not found in either database, SQL Server sends an error message to the error log.
- SQL Server uses a **hysteresis value**, the global variable *@@thresh_hysteresis*, to determine how sensitive thresholds are to variations in free space. Once a threshold executes its procedure, it is deactivated. The threshold remains inactive until the amount of free space in the segment rises to *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space.

The Last-Chance Threshold

- By default, SQL Server monitors the free space on the segment where the log resides and executes **sp_thresholdaction** when the amount of free space is less than that required to permit a successful dump of the transaction log. This amount of free space, the “last-chance threshold”, is calculated by SQL Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, SQL Server suspends the transaction until log space is freed. Use **sp_dboption** to change this behavior for a particular database. Setting the **abort tran on log full** option to **true** causes SQL

Server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.

- You cannot use **sp_modifythreshold** to change the free-space value or segment name associated with the last-chance threshold.

Other Thresholds

- Each database can have up to 256 thresholds, including the last-chance threshold.
- Each threshold must be at least 2 times *@@thresh_hysteresis* pages from the next closest threshold.
- Use **sp_helpthreshold** for information about existing thresholds.
- Use **sp_dropthreshold** to drop a threshold from a segment.

Creating Threshold Procedures

- Any user with **create procedure** permission can create a threshold procedure in a database. Usually, a System Administrator creates **sp_thresholdaction** in the *master* database, and Database Owners create threshold procedures in user databases.
- **sp_modifythreshold** does not verify that the specified procedure exists. It is possible to associate a threshold with a procedure that does not yet exist.
- SQL Server passes four parameters to a threshold procedure:
  - *@dbname*, *varchar*(30), which identifies the database
  - *@segmentname*, *varchar*(30), which identifies the segment
  - *@space_left*, *int*, which indicates the number of free pages associated with the threshold
  - *@status*, *int*, which has a value of 1 for last-chance thresholds and 0 for other thresholds

  These parameters are passed by position rather than by name; your threshold procedure can use other names for them, but must declare them in the order shown and with the correct datatypes.
- It is not necessary to create a different procedure for each threshold. To minimize maintenance, create a single threshold procedure in the *sybsystemprocs* database that all thresholds on the SQL Server execute.

- Include **print** and **raiserror** statements in the threshold procedure to send output to the error log.

Executing Threshold Procedures

- Tasks initiated when a threshold is crossed execute as background tasks. These tasks do not have an associated terminal or user session. If you execute **sp_who** while these tasks are running, the *status* column shows "background".
- SQL Server executes the threshold procedure with the permissions of the user who modified the threshold, at the time he or she executed **sp_modifythreshold**, minus any permissions that have since been revoked.
- Each threshold procedure uses one user connection, for as long as it takes to execute the procedure.

Disabling Free-Space Accounting

- Use the **no free space acctg** option of **sp_dboption** to disable free-space accounting on non-log segments.
- You cannot disable free-space accounting on log segments.

◆ ***WARNING!***

System procedures cannot provide accurate information about space allocation when free-space accounting is disabled.

Creating Last-Chance Thresholds for Pre-Release 10.0 Databases

- When you upgrade an existing database to Release 10.0, it does not automatically acquire a last-chance threshold.
- Use the **lct_admin** system function to create a last-chance threshold in an existing database.
- Only databases that store their logs on a separate segment can have a last-chance threshold. Use **sp_logdevice** to move the transaction log to a separate device.

Messages

- `This procedure can only affect thresholds in the current database. Say "USE` *`database_name`*`",then run this procedure again.`

  sp_modifythreshold can modify thresholds only in the database you are currently using. Issue the **use** command to open the database in which you want to modify a threshold. Then run **sp_modifythreshold** again.

- `You may not alter the free space or segment name of the log's last-chance threshold.`

  sp_modifythreshold cannot change the free-space value or segment name associated with the last-chance threshold.

Permissions

Only the Database Owner or a System Administrator can execute **sp_modifythreshold**.

Tables Used

master..sysusages, *sysobjects*, *syssegments*, *systhresholds*

See Also

Commands	**create procedure, dump transaction**
System procedures	**sp_addthreshold, sp_dboption, sp_dropthreshold, sp_helpthreshold, sp_thresholdaction**

sp_monitor

Function

Displays statistics about SQL Server.

Syntax

sp_monitor

Parameters

None.

Examples

```
        sp_monitor
last_run                current_run         seconds
-------------------     ------------------- ---------
Jan 29 1987 10:11AM     Jan 29 1987 10:17AM 314

cpu_busy            io_busy      idle
---------------     ---------    --------------
4250(215)-68%       67(1)-0%     109(100)-31%

packets_received        packets_sent     packet_errors
----------------        ------------     ------------
781(15)                 10110(9596)      0(0)

total_read      total_write total_errors     connections
-----------     ------------------------     -----------
394(67)         5392(53)    0(0)             15(1)
```

Reports information about how busy SQL Server has been.

Comments

- SQL Server keeps track of how much work it has done in a series of global variables. **sp_monitor** displays the current values of these global variables, and how much they have changed since the last time the procedure executed.
- For each column, the statistic appears in the form *number(number)-number%* or *number(number)*. The first number refers to the number of seconds (for *cpu_busy*, *io_busy*, and *idle*) or the total number (for the other variables) since SQL Server restarted. The number in parentheses refers to the number of seconds or total number since the last time **sp_monitor** ran. The percentage is the percent of time since **sp_monitor** last ran.

For example, if the report showed *cpu_busy* as 4250(215)-68%, this would mean that the CPU was busy 4250 seconds since SQL Server last started up, 215 seconds since **sp_monitor** last ran, and 68% of the total time since **sp_monitor** last ran.

For the *total_read* variable, the value 394(67) means there have been 394 disk reads since SQL Server was restarted, 67 of them since the last time **sp_monitor** was run.

- Table 1-14 describes the columns in the **sp_monitor** report, the equivalent global variables, if any, and their meanings. With the exception of *last_run*, *current_run* and *seconds*, these column headings are also the names of global variables—except that all global variables are preceded by @@. There is also a difference in the units of the numbers reported by the global variables—the numbers reported by the global variables are not milliseconds of CPU time, but machine ticks.

Column	Variable	Meaning
last_run		The clock time at which the **sp_monitor** procedure last ran.
current_run		The current clock time.
seconds		The number of seconds since **sp_monitor** last ran.
cpu_busy	*@@cpu_busy*	The number of seconds in CPU time that SQL Server's CPU was doing SQL Server work.
io_busy	*@@io_busy*	The number of seconds in CPU time that SQL Server has spent doing input and output operations.
idle	*@@idle*	The number of seconds in CPU time that SQL Server has been idle.
pack_received	*@@pack_received*	The number of input packets read by SQL Server.
pack_sent	*@@pack_sent*	The number of output packets written by SQL Server.
packet_errors	*@@packet_errors*	The number of errors detected by SQL Server while reading and writing packets.
total_read	*@@total_read*	The number of disk reads by SQL Server.
total_write	*@@total_write*	The number of disk writes by SQL Server.
total_errors	*@@total_errors*	The number of errors detected by SQL Server while reading and writing.
connections	*@@connections*	The number of logins or attempted logins to SQL Server.

Table 1-14: Columns in the sp_monitor Report

Messages

- `Can’t run sp_monitor from within a transaction.`

  sp_monitor modifies system tables, so it cannot be run within a transaction.

Permissions

Only a System Administrator can execute **sp_monitor**.

Tables Used

master.dbo.sysengines, *master.dbo.spt_monitor*, *sysobjects*

See Also

System procedures	**sp_who**
Topics	Variables (Local and Global)

sp_password

Function

Adds or changes a password for a SQL Server login account.

Syntax

```
sp_password caller_passwd, new_passwd [, login_name]
```

Parameters

caller_passwd – is your password. When you are changing your own password, this is your old password. When a System Security Officer is using **sp_password** to change another user's password, *caller_passwd* is the System Security Officer's password.

new_passwd – is the new password for the user, or for *login_name*. It must be at least 6 bytes long. Enclose passwords that include characters besides A-Z, a-z, or 0-9 in quotation marks. Also enclose passwords that begin with 0-9 in quotes.

login_name – the login name of the user whose account password the System Security Officer is changing.

Examples

1. `sp_password "3blindmice, "2mediumhot"`

   Changes your password from password from "3blindmice" to "2mediumhot." (Enclose the passwords in quotes because they begin with numerals.)

2. `sp_password “2tomato”, sesame1, victoria`

   A System Security Officer whose password is "2tomato" has changed Victoria's password to "sesame1."

3. `sp_password null, "16tons"`

   Changes your password from NULL to "16tons." Notice that NULL is not enclosed in quotes. (NULL is not a permissible new password.)

4. `PRODUCTION...sp_password figaro, lilacs`

   Changes your password on the PRODUCTION server from "figaro" to "lilacs."

Comments

- Any user can change his or her password with **sp_password**.
- Only a System Security Officer can use the *login_name* parameter to change another user's password.
- New passwords must be at least 6 bytes long. They cannot be NULL.
- The encrypted text of *caller_passwd* must match the existing encrypted password of the caller. If it does not, **sp_password** returns an error message and fails. *master.dbo.syslogins* lists passwords in encrypted form.
- If a client program requires users to have the same password on remote servers as on the local server, users must change their passwords on all the remote servers before changing their local passwords. Execute **sp_password** as a remote procedure call on each remote server. See Example 4.
- You can set the **passwordexp** configuration variable to establish a password expiration interval that forces all SQL Server login accounts to change passwords on a regular basis. See the *System Administration Guide* for information.

Messages

- `Can't run sp_password from within a transaction.`

  sp_password modifies system tables, so it cannot be run within a transaction.
- `Error: Unable to set the Password.`

  Check your syntax carefully and try again to set the password.
- `No such login -- no password changed.`

  The name supplied for the *login_name* parameter does not exist on SQL Server.
- `Invalid caller's password specified, password left unchanged.`

  The *caller_passwd* parameter is not the current password of the caller.
- `New password specified is too short. Minimum length of acceptable passwords is 6 characters.`

  You specified a password that is too short.

- `New password supplied is the same as previous password. Please supply a different password.`

  If *new_passwd* is at least six bytes long, it is encrypted and compared with the encrypted value of *login_name*'s existing encrypted password. If they differ, the encrypted text of *new_passwd* is saved; otherwise **sp_password** fails and returns this message.

- `Password correctly set.`

  The password was successfully changed. Use the new password the next time you log in to SQL Server.

Permissions

Any user can execute **sp_password** to change his or her own password. Only a System Security Officer can use **sp_password** to change other users' passwords.

Tables Used

master.dbo.syslogins, sysobjects

See Also

System procedures	**sp_addlogin, sp_adduser**
Topics	Roles, Login Management

sp_placeobject

Function

Puts future space allocations for a table or index on a particular segment.

Syntax

sp_placeobject *segname*, *objname*

Parameters

segname – is the name of the segment on which to locate the table or index.

objname – is the name of the table or index for which to place subsequent space allocation on the segment *segname*.

Examples

1. **sp_placeobject indexes, 'employee.employee_nc'**

   This command places all subsequent space allocation for the index named *employee_nc* on table *employee* on the segment named *indexes*.

Comments

- You cannot change the location of future space allocations for system tables.
- Placing a table or index on a particular segment does not affect the location of any existing table or index data. It only affects future space allocation. Changing the segment used by a table or index can spread the data among multiple segments.
- You can specify a segment if you create a table or index with **create table** or **create index**. If you do not specify a segment, the data goes on the *default* segment.
- When **sp_placeobject** splits a table or index across more than one disk fragment, the diagnostic command **dbcc** displays messages about the data that resides on fragments that were in use for storage before **sp_placeobject** executed. Ignore these messages.

Messages

- `'`*`objname`*`' is now on segment '`*`segname`*`'.`

  The command was successful. Data for the *objname* is now put onto the segment *segname*.

- `There is no index named '`*`indexname`*`' for table '`*`tablename`*`'.`

  The index referenced in the *objname* parameter does not exist. Use the system procedure **sp_helpindex** to list a table's indexes.

- `There is no such segment as `*`segname`*`.`

  The *segname* you have referenced is not a segment. All segments for a database are listed in the *syssegments* table. Use **sp_helpsegment** to get a report on all segments.

- `There is no table named '`*`tablename`*`'.`

  The table referenced in the *objname* parameter does not exist. Use the system procedure **sp_help** for a list of existing tables.

- `You do not own table '`*`tablename`*`'.`

  Only the table owner, the Database Owner, or a System Administrator can place a table or its index on a segment.

- `Use sp_logdevice to move syslogs table.`
- `You can't move system tables.`

  System tables must remain on the *system* segment.

Permissions

Only the table owner, Database Owner, or a System Administrator can execute **sp_placeobject**.

Tables Used

sysindexes, sysobjects, syssegments

See Also

Commands	**dbcc**
System procedures	**sp_addsegment, sp_dropsegment, sp_extendsegment, sp_help, sp_helpindex, sp_helpsegment**

sp_primarykey

Function

Defines a primary key on a table or view.

Syntax

sp_primarykey *tabname*, *col1* [, *col2*, *col3*, ..., *col8*]

Parameters

tabname – is the name of the table or view on which to define the primary key.

col1 – is the name of the first column that makes up the primary key. The primary key can consist of one to eight columns.

Examples

1. **sp_primarykey authors, au_id**

   Defines the *au_id* field as the primary key of the table *authors*.

2. **sp_primarykey employees, lastname, firstname**

   Defines the combination of the fields *lastname* and *firstname* as the primary key of the table *employees*.

Comments

- Executing **sp_primarykey** adds the key to the *syskeys* table. Only the owner of a table or view can define its primary key.
- Create keys to make explicit a logical relationship that is implicit in your database design. An application program can use the information.
- A table or view can have only one primary key. To display a report on the keys that have been defined, execute **sp_helpkey.**
- The installation process runs **sp_primarykey** on the appropriate columns of the system tables.

Messages

- `New primary key added.`

  You successfully defined a new primary key.

- `The table or view named doesn't exist in the current database.`

  The table or view supplied as the *tabname* parameter doesn't exist in the current database.

- `Only the owner of the table may define a primary key.`

  You aren't the owner of the table or view and, therefore, cannot define its primary key.

- `Primary key already exists on table -- drop key first.`

  A table or view can have only have one primary key; one already exists on the table or view supplied as the *tabname* parameter.

- `Table or view name must be in the current database.`

  You can only define a primary key for a table in the current database.

- `The table has no such` *`nth`* `column.`

  The column name supplied as one of the column names isn't a column in *tabname*.

Permissions

Only the owner of a table or view can execute **sp_primarykey**.

Tables Used

syscolumns, *syskeys*, *sysobjects*

See Also

Commands	**create trigger**
System procedures	**sp_commonkey, sp_dropkey, sp_foreignkey, sp_helpjoins, sp_helpkey**

sp_procxmode

Function

Displays or changes the transaction modes associated with stored procedures.

Syntax

sp_procxmode [*procedure_name* [, *transaction_mode*]]

Parameters

procedure_name – is the name of the stored procedure whose transaction mode you are examining or changing.

transaction_mode – is the new transaction mode for the stored procedure. The valid values are “chained”, “unchained”, and “anymode”.

Examples

1. **sp_procxmode**

```
procedure name       user name   transaction mode
------------------   ---------   ----------------
byroyalty            dbo         Unchained
discount_proc        dbo         Unchained
history_proc         dbo         Unchained
insert_sales_proc    dbo         Unchained
insert_detail_proc   dbo         Unchained
storeid_proc         dbo         Unchained
storename_proc       dbo         Unchained
title_proc           dbo         Unchained
titleid_proc         dbo         Unchained
```

   Displays the transaction mode for all stored procedures in the current database.

2. **sp_procxmode byroyalty**

```
procedure name                   transaction mode
------------------------------   ----------------
byroyalty                        Unchained
```

   Displays the transaction mode of the stored procedure *byroyalty*.

3. **sp_procxmode byroyalty, "chained"**

   Changes the transaction mode for the stored procedure *byroyalty* in the *pubs2* database from “unchained” to “chained”.

Comments

- You must be the owner of the stored procedure, the owner of the database which contains the stored procedure, or the System Administrator in order to change the transaction mode of a stored procedure. The Database Owner or System Administrator can change the mode of another user's stored procedure by qualifying it with the database and user name. For example:

  ```
  sp_procxmode "otherdb.otheruser.newproc", "chained"
  ```

- To use **sp_procxmode**, turn off chained transaction mode using the **chained** option of the **set** command. By default, this option is turned off.
- When you use **sp_procxmode** with no parameters, it reports the transaction modes of every stored procedure in the current database.
- To examine a stored procedure's transaction mode (without changing it), enter:

  `sp_procxmode` ***`procedure_name`***

- To change a stored procedure's transaction mode, enter:

  `sp_procxmode` ***`procedure_name, transaction_mode`***

- When you create a stored procedure, SQL Server tags it with the current session's transaction mode. This means:
  - You can execute "chained" stored procedures only in sessions using chained transaction modes.
  - You can execute "unchained" stored procedures only in sessions using unchained transaction mode.

  To execute a particular stored procedure in either chained or unchained sessions, set its transaction mode to "anymode".
- If you attempt to run a stored procedure under the wrong transaction mode, SQL Server returns a warning message, but the current transaction, if any, is not affected.

Messages

- `The new transaction-mode must be unchained, chained, or anymode.`

  You specified and invalid mode.
- `The specified object is not a stored procedure in the current database.`

  You specified an invalid object name.

- `You must be either the system administrator (SA), the database administrator (dbo), or the owner of this stored procedure to change its transaction mode.`

  You do not have the correct permissions.

- `You cannot change the mode of a remote stored procedure.`

Permissions

Only a System Administrator, the Database Owner, or the owner of the procedure can change its transaction mode.

Tables Used

#tranmode, sysobjects

See Also

Commands	**begin transaction, commit transaction, rollback transaction, set**
Topics	Transactions, Variables

sp_recompile

Function

Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.

Syntax

```
sp_recompile tabname
```

Parameters

tabname – is the name of a table in the current database.

Examples

1. **`sp_recompile titles`**

   Recompiles each trigger and stored procedure that uses the table *titles* the next time the trigger or stored procedure runs.

Comments

- The queries used by stored procedures and triggers are optimized only once, when they are compiled. As you add indexes or make other changes to your database that affect its statistics, your compiled stored procedures and triggers may lose efficiency. By recompiling the stored procedures and triggers that act on a table, you can optimize the queries for greatest efficiency.
- **sp_recompile** looks for *tabname* in the current database only.
- You cannot use **sp_recompile** on system tables.

Messages

- `Object 'objname' is not a table.`

  The specified table does not exist in the current database.
- `Table or view name must be in current database.`

  You can only use **sp_recompile** on objects in the current database.

-
  ```
  'tabname' is a system table. Cannot use
  sp_recompile on system tables.
  ```

  sp_recompile is only allowed on user tables.

- `You do not own table` *`tabname`*`.`

  You can only use **sp_recompile** on tables that you own. If you are a System Administrator, you can run **sp_recompile** on any table.

-
  ```
  Each stored procedure and trigger that uses table
  'tabname' will be recompiled the next time it is
  executed.
  ```

  sp_recompile ran successfully. All stored procedures and triggers that use the named table recompile the next time they run.

Permissions

Any user can execute **sp_recompile**.

Tables Used

sysobjects

See Also

Commands	**create index**

sp_remap

Function

Remaps a Release 4.8 or later stored procedure, trigger, rule, default, or view to be compatible with Release 10.0. Use **sp_remap** on objects that the Release 10.0 upgrade procedure failed to remap.

Syntax

```
sp_remap object_name
```

Parameters

object_name – is the name of a stored procedure, trigger, rule, default, or view in the current database.

Examples

1. **`sp_remap myproc`**

   Remaps a stored procedure called *myproc*.

2. **`sp_remap "my_db..default_date"`**

   Remaps a rule called *default_date*. Execute a **use** statement to open the correct database before running **sp_remap**.

Comments

- If **sp_remap** fails to remap an object, drop the object from the database and recreate it. Before running **sp_remap** on an object, it is a good idea to copy its definition into an operating system file with the **defncopy** utility. For more information about **defncopy**, see the *SQL Server Utility Programs* for your operating system.
- **sp_remap** can cause your transaction log to fill rapidly. Before running **sp_remap**, use the **dump transaction** command to dump the transaction log, as needed.
- Only a System Administrator or the owner of an object or can remap the object with **sp_remap**.
- You can use **sp_remap** only on objects in the current database.
- **sp_remap** makes no changes to objects that have already been upgraded to Release 10.0.

Messages

- `Object does not exist in this database.`

  The object you tried to remap does not exist in the current database. Issue a **use** *database* statement to open the correct database, then re-execute **sp_remap**.

- `DBCC execution completed. If DBCC printed error messages, contact a user with System Administrator (SA) authorization.`

  sp_remap executed the **remap** option of the **dbcc** command.

- `You do not own object` *`object_name`*`.`

  Only the owner of an object can remap it.

- `Remapping utility - procedure is corrupted in Sysprocedures. Recreate this procedure.`

  sp_remap cannot remap this object. Drop the object from the database and recreate it.

- `Remapping utility - a pointer exists in a tree when it should not.`

  sp_remap cannot remap this object. Drop the object from the database and recreate it.

- `Remapping utility - unable to locate the given procedure` *`procedure_name`* `in Sysprocedures.`

  sp_remap cannot remap this object. Drop the object from the database and recreate it.

- `Remapping utility -- Procedure needs to be recreated for this port.`

  sp_remap cannot remap this object. Drop the object from the database and recreate it.

Permissions

Any user can execute **sp_remap.**

Tables Used

master.dbo.sysdatabases, sysobjects

See Also

Commands	**create default, create procedure, create rule, create trigger, create view, drop default, drop procedure, drop rule, drop trigger, drop view, dump transaction**
Utility Programs	**defncopy**

sp_remoteoption

Function

Displays or changes remote login options.

Syntax

```
sp_remoteoption [remote_server, login_name,
   remote_name, opt_name, {true | false}]
```

Parameters

remote_server – is the name of the remote server that has the remote login to which to apply the option.

login_name – is the login name that identifies the remote login for the *remote_server, login_name, remote_name* combination.

remote_name – is the remote user name that identifies the remote login for the *remote_server, login_name, remote_name* combination.

opt_name – is the name of the option you want to turn on or off. Currently there is only one option, **trusted**, which means that the local server accepts remote logins from other servers without user-access verification for the particular remote login. The default is to use password verification. SQL Server understands any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

true | **false** – **true** turns the option on, **false** turns it off.

Examples

1. **sp_remoteoption**

   ```
   Settable remote login options.
   remotelogin_option
   -----------------------
   trusted
   ```

   Displays a list of the remote login options.

2. **sp_remoteoption GATEWAY, churchy, pogo, trusted, true**

   Defines the remote login from the remote server GATEWAY to be **trusted** (that is, the password is not checked).

3. **`sp_remoteoption GATEWAY, churchy, pogo, trusted, false`**

   Defines the remote login from the remote server GATEWAY to be untrusted (that is, the password is checked).

Comments

- To display a list of the remote login options, execute **sp_remoteoption** with no parameters.
- See the *System Administration Guide* for additional details on remote login options.

Messages

- `Option 'opt_name' turned off.`

  The procedure was successful.

- `Option 'opt_name' turned on.`

  The procedure was successful.

-
  ```
  Remote login option doesn't exist or can't be set
  by user.
  Run sp_remoteoption with no parameters to see
  options.
  ```

  Either the option doesn't exist, or the user does not have permission to turn it on or off.

- `Remote login option is not unique.`

  The name supplied as the *opt_name* parameter is not unique. No remote login option value was changed. The complete names that match the string supplied appear, so you can see how to make *opt_name* more specific.

- `Settable remote login options.`

  Executing **sp_remoteoption** with no parameters displays a list of options the user can set. (See Example 1.)

-
  ```
  There is no remote user 'remote_name' mapped to
  local user 'login_name' from the remote server
  'remote_server'.
  ```

  You incorrectly identified the remote login or the remote server name. Run **sp_helpremotelogin** to list the remote logins. Run **sp_helpserver** to list the remote servers.

- `Usage: sp_remoteoption [remoteserver, loginame, remotename, optname, {true | false}]`

  Either the *opt_name* parameter was omitted or the *opt_value* parameter not **true** or **false**.

Permissions

Only System Security Officers can execute **sp_remoteoption**.

Tables Used

master.dbo.spt_values, *master.dbo.sysmessages*, *master.dbo.sysremotelogins*, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_helpremotelogin**

sp_rename

Function

Changes the name of a user-created object in the current database.

Syntax

```
sp_rename objname, newname
```

Parameters

objname – is the original name of the user-created object (table, view, column, stored procedure, index, trigger, default, rule, check constraint, or referential constraint) or datatype. If the object to rename is a column in a table, *objname* must be in the form *"table.column"*.

You can only rename an object in the current database, and only if you own it. This rule holds for the Database Owner and System Administrator as well as for other users.

newname – is the new name of the object or datatype. Names of objects and datatypes must conform to the rules for identifiers and be unique to the current database.

Examples

1. **sp_rename titles, books**

   Renames the *titles* table to *books*.

2. **sp_rename "books.title", bookname**

   Renames the *title* column in the *books* table to *bookname*.

3. **sp_rename tid, bookid**

   Renames the user-defined datatype *tid* to *bookid*.

Comments

- **sp_rename** changes the name of a user-created object or datatype. You can only change the name of an object or datatype in the current database.
- When you are renaming a column, you must leave off the table name prefix from the new column name or SQL Server does not accept the new name. See Example 2.

- You cannot change the names of system objects and system datatypes.

◆ ***WARNING!***

Procedures, triggers, and views that depend on an object whose name has been changed work until they are recompiled. Recompilation takes place for many reasons, and without notification to the user. Also, the old object name appears in query results until the user changes and recompiles the procedure, trigger, or view. Change the definitions of any dependent objects when you execute sp_rename. Find dependent objects with sp_depends.

Messages

- `Column name has been changed.`

  The specified column name was renamed to *newname*.
- `Index name has been changed.`

  The specified index name was renamed to *newname*.
- `Name of user-defined type name changed.`

  The specified user-defined datatype was renamed to *newname*.
- `Newname already exists in sysobjects.`

  The object named in *newname* already exists. Object names must be unique to the database.
- `Newname already exists in systypes.`

  The datatype named in *newname* already exists. Datatype names must be unique to the database.
- *`newname`* `is not a valid name.`

  newname does not conform to the rules for identifiers.
- `Object must be in the current database.`

  The name supplied for the *objname* parameter included a reference to a database. The object must be in the current database.
- `Object name cannot be changed either because it does not exist in this database, or you don’t own it, or it is a system name.`

  No object of the specified name exists, or you don’t own the object.

- `Object name has been changed.`

  The specified object was renamed to *newname*.

- `There is already a column named 'newname' in table 'tablename'.`

  Column names must be unique within a table. The table already contains a column with the name you chose.

- `Table or view names beginning with '#' are not allowed.`

  You cannot begin the name of a table or view with "#".

- `There is already an index named 'newname' for table 'tablename'.`

  Index names for a table must be unique. The table already has an index with the name you chose.

- `You do not own a table, column or index of that name in the current database.`

  No column of the specified name exists in the specified table, or you don't own the table.

Permissions

Users can execute **sp_rename** for their own objects. Only the Database Owner and a System Administrator can execute **sp_rename** for all objects.

Tables Used

syscolumns, sysindexes, sysobjects, systypes

See Also

Commands	**alter table, create default, create procedure, create rule, create table, create trigger, create view**
System procedures	**sp_addtype, sp_checreswords, sp_depends, sp_renamedb**
Topics	Datatypes

sp_renamedb

Function

Changes the name of a database. You **cannot** rename system databases or databases with external referential integrity constraints.

Syntax

`sp_renamedb` ***`dbname`***, ***`newname`***

Parameters

dbname – is the original name of the database.

newname – is the new name of the database. Database names must conform to the rules for identifiers and must be unique.

Examples

1. **`sp_renamedb accounting, financial`**

   Renames the *accounting* database to *financial*.

Comments

- Executing **sp_renamedb** changes the name of a database.
- The System Administrator must place a database in single-user mode with **sp_dboption** before renaming it, and restore it to multi-user mode afterwards.
- **sp_renamedb** fails if any table in the database references, or is referenced by, a table in another database. Use 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
  ```

  Use 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
  ```

Use **alter table** to drop the cross-database constraints in these tables, then rerun **sp_renamedb**.

◆ ***WARNING!***

Procedures, triggers, and views that depend on a database whose name has been changed work until they are recompiled. Recompilation takes place for many reasons, and without notification to the user. When SQL Server recompiles the procedure, trigger, or view, it no longer works. Change the definitions of any dependent objects when you execute sp_renamedb. Find dependent objects with sp_depends.

Messages

- `A database with the new name already exists.`

  The database you specified for the *newname* parameter is already a database. Database names must be unique.

- `Can’t run sp_renamedb from within a transaction.`

  sp_renamedb modifies system tables, so it cannot be run within a transaction.

- `Database '`*`database_name`*`' has references to other databases. Drop those references and try again.`

  You cannot rename a database if any of its tables references—or is referenced by—a table in another database. Before renaming the database, you must use **alter table** to drop any external referential integrity constraints.

- `Database is renamed and in single-user mode. System Administrator (SA) must reset it to multi-user mode with sp_dboption.`

  sp_renamedb succeeded.

- *`newname`*` is not a valid name.`

  The value for *newname* does not conform to the rules for identifiers.

- `The databases 'master', 'model', and 'tempdb' cannot be renamed.`

  You cannot rename system databases.

- `The specified database does not exist.`

  The database you specified with the *dbname* parameter doesn’t exist.

- `System Administrator (SA) must set database '`*`dbname`*`' to single-user mode with sp_dboption before using sp_renamedb.`

  You can't rename a database while someone is using it, and you must make sure that no one tries to use the database while it is being renamed.

Permissions

Only System Administrators can execute **sp_renamedb**.

Tables Used

master.dbo.spt_values, master.dbo.sysdatabases, sysobjects

See Also

Commands	**create database**
System procedures	**sp_changedbowner, sp_dboption, sp_depends, sp_helpdb, sp_rename**

sp_reportstats

Function

Reports statistics on system usage.

Syntax

```
sp_reportstats [user_name]
```

Parameters

user_name – is the login name of a user to show accounting totals for.

Examples

1. **sp_reportstats**

```
Name    Since        CPU     Percent CPU   I/O     Percent I/O
------  -----------  -----   ------------  -----   -------------
probe   jun 19 1993  0       0%            0       0%
julie   jun 19 1993  10000   24.9962%      5000    24.325%
jason   jun 19 1993  10002   25.0013%      5321    25.8866%
ken     jun 19 1993  10001   24.9987%      5123    24.9234%
kathy   jun 19 1993  10003   25.0038%      5111    24.865%

(5 rows affected)
              Total CPU   Total I/O
              ---------   ---------
              40006       20555

              (1 row affected, return status = 0)
```

Displays a report of current accounting totals for all SQL Server users.

2. **sp_reportstats kathy**

```
Name    Since        CPU     Percent CPU   I/O     Percent I/O
------  -----------  -----   ------------  -----   -------------
kathy   Jul 24 1993  498     49.8998%      48392   9.1829%

(1 row affected)
              Total CPU   Total I/O
              ---------   ----------
              998         98392

              (1 row affected, return status = 0)
```

Displays a report of current accounting totals for user “kathy.”

Comments

- **sp_reportstats** prints out the current accounting totals for all logins, as well as each login's individual statistics and percentage of the overall statistics. Statistics for any process with an *suid* of 1—*sa*, checkpoint, network, and mirror handlers—are not recorded.
- **sp_reportstats** accepts one parameter, the login name of the account to report. With no parameters, **sp_reportstats** reports on all accounts.

Messages

- `No login with the specified name exists.`

  Check the spelling of the user's name.

Permissions

Only System Administrators can execute **sp_reportstats**.

Tables Used

master.dbo.syslogins, sysobjects

See Also

System procedures	**sp_clearstats, sp_configure**

sp_role

Function

Grants or revokes roles to a SQL Server login account.

Syntax

```
sp_role {"grant" | "revoke"},
   {sa_role | sso_role | oper_role}, login_name
```

Parameters

grant | **revoke** – specifies whether to grant the role to or revoke the role from *login_name*.

sa_role | **sso_role** | **oper_role** – is the level of role to grant or revoke.

login_name – is the login account to which to grant or revoke the role.

Examples

1. **sp_role "grant", sa_role, alexander**

   Grants the System Administrator role to the login account named “alexander”.

Comments

- You cannot revoke the System Security Officer role from the server’s last remaining System Security Officer account. Similarly, you cannot revoke the System Administrator role from the last remaining System Administrator account.
- When you grant a role to a user, it takes effect the next time the user logs into SQL Server. However, the user can immediately enable the role by using the **set role** command. For example, the following command:

  ```
  set role "sa_role" on
  ```

  enables the System Administrator role for the user.
- You cannot revoke a role from a user while the user is logged in.
- When users log in, all roles that have been granted to them are automatically active. To turn off a role, use the **set** command. For example, to deactivate the System Administrator role, use the following command:

  ```
  set role "sa_role" off
  ```

Messages

- `Can’t run sp_role from within a transaction.`

  sp_role modifies system tables, so it cannot be run within a transaction.

- `No such account -- nothing changed.`

  The login name you specified does not exist.

- `Invalid role -- nothing changed.`

  You specified a role that does not exist.

- `Cannot revoke SSO or SA role from the last remaining unlocked SSO or SA login.`

  There must always be at least one unlocked System Security Officer and System Administrator account.

- `Neither 'grant' nor 'revoke' is specified -- nothing changed.`

  Specify either **grant** or **revoke**.

- `Role updated.`

  sp_role successfully executed.

- `Warning: the specified account is active.`

  You cannot revoke a role from a user who is currently logged in.

Permissions

Only a System Administrator can grant the System Administrator role to other users. Only a System Security Officer can grant the System Security Officer or Operator role to other users.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, master.dbo.syssrvroles, sysobjects

See Also

Commands	**grant, revoke, set**
System procedures	**sp_displaylogin**
Topics	Roles, System Functions

sp_serveroption

Function

Displays or changes remote server options.

Syntax

sp_serveroption [*server*, *optname*, {true | false}]

Parameters

server – is the name of the remote server for which to set the option.

optname – is the name of the option to set or unset. Currently there are two options: **net password encryption** and **timeouts**.

Option	Meaning
net password encryption	Specifies whether to initiate connections with a remote server with the client side password encryption handshake or with the normal (unencrypted password) handshake sequence. The default is “false”, no network encryption.
timeouts	When unset (“false”), disables the normal timeout code used by the local server, so the site connection handler does not automatically drop the physical connection after one minute with no logical connection. The default is “true”.

Table 1-15: sp_serveroption Options

SQL Server understands any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

true | **false** – **true** sets the option, **false** unsets the option.

Examples

1. **`sp_serveroption`**

   ```
   Settable server options.
   server_option
   ------------------------
   timeouts
   net password encryption
   ```

   Displays a list of the server options.

2. **`sp_serveroption GATEWAY, "timeouts", false`**

   Tells the server not to time out inactive physical connections with the remote server GATEWAY.

3. **`sp_serveroption GATEWAY,`**
   `    "net password encryption", true`

   Specifies that when making connections to the remote server GATEWAY, GATEWAY sends back an encryption key to encrypt the password to send to it.

Comments

- To display a list of the user-settable server options, execute **sp_serveroption** with no parameters.
- Once **timeouts** is set to "false," the site handlers continue to run until one of the two servers is shut down.
- The **net password encryption** option allows clients to specify whether to send passwords in plain text or encrypted form over the network when a initiating a remote procedure call. If **net password encryption** is set to "true," the initial login packet is sent without passwords, and the client indicates to the remote server that encryption is desired. The remote server sends back an encryption key, which the client uses to encrypt its passwords. The client then encrypts its passwords, and the remote server uses the key to authenticate them when they arrive.
- To set network password encryption for a particular **isql** session, you can use the UNIX **-X** or OpenVMS **/encrypt** option to **isql**. See the *SQL Server Utility Programs* manual for more information.
- The **net password encryption** option works only between SQL Servers of release 10.0 and later.
- See the *System Administration Guide* for additional details on server options.

Messages

- `Can’t run sp_serveroption from within a transaction.`

  sp_serveroption modifies system tables, so it cannot be run within a transaction.

- `No such server -- run sp_helpserver to list servers.`

  You specified an incorrect server name. Run **sp_helpserver** to get a list of servers.

- `Option can be set for remote servers only -- not the local server.`

  You tried to set an option on the local server.

- `Server option doesn’t exist or can’t be set by user. Run sp_serveroption with no parameters to see options.`

  Either the option doesn’t exist, or you do not have permission to set or unset it. Run **sp_serveroption** with no parameters to display a list of settable options.

- `Server option is not unique.`

  The name supplied as the *optname* parameter is not unique. No server option value was changed.

- `Usage: sp_serveroption [server, optname, {true | false}]`

  Either the *optname* parameter was omitted or the third parameter was not **true** or **false**.

Permissions

Any user can execute **sp_serveroption** with no parameters to generate a list of the options. Only System Administrators can use the **timeouts** option. Only System Security Officers can use the **net password encryption** option.

Tables Used

master.dbo.sysservers, *sysobjects*

See Also

System procedures	**sp_helpserver, sp_password**
Topics	Login Management

sp_setlangalias

Function

Assigns or changes the alias for an alternate language.

Syntax

sp_setlangalias ***language*****,** ***alias***

Parameters

language – is the official language name of the alternate language.

alias – is the new local alias for the alternate language.

Examples

1. **sp_setlangalias french, francais**

   This command assigns the alias name "francais" for the official language name french.

Comments

- *alias* replaces the current value of *syslanguages.alias* for official name *language*.
- The **set language** command can use the new *alias* in place of the official language name.

Messages

- `*language* is not an official language name from syslanguages.`

  Use **sp_helplanguage** to see a list of official names of alternate languages on this SQL Server.
- `*alias* alias already exists in syslanguages.`

  The new *alias* must be unique. Use **sp_helplanguage** to see a list of official names and aliases available on this SQL Server.
- `Language alias not changed.`

  An error occurred while updating *master.dbo.syslanguages*, so the alias was not added. The SQL Server message that appeared before this message provides more information.
- `Language alias reset.`

  The alias for this alternate language name was changed.

Permissions

System Administrators can execute **sp_setlangalias**, and can grant permission to others.

Tables Used

master.dbo.syslanguages, *sysobjects*

See Also

Commands	**set**
System procedures	**sp_addlanguage, sp_droplanguage, sp_helplanguage**

sp_spaceused

Function

Displays the number of rows, the number of data pages, and the space used by one table or by all tables in the current database.

Syntax

sp_spaceused [*tablename*]

Parameters

tablename – is the name of the table on which to report. If omitted, a summary of space used in the current database appears.

Examples

1. **sp_spaceused titles**

```
name       rowtotal   reserved   data    index_size    unused
---------- ---------  ---------  ------- ----------    ----------
titles     18         46 KB      6 KB    4 KB          36 KB
```

Reports on the amount of space allocated (reserved) for the *titles* table, the amount used for data, the amount used for index(es), and the available (unused) space.

2. **sp_spaceused**

```
database_name   database_size
--------------- ---------------------------
master          5 MB

reserved     data         index_size  unused
---------    ---------    ----------- --------
2176 KB      1374 KB      72 KB       730 KB
```

Prints a summary of space used in the current database.

Comments

- **sp_spaceused** computes the number of data pages, and the space used by an object or by each object in the current database. The number of rows, reported as *rowtotal*, is an estimate.
- **sp_spaceused** reports only on the amount of space affected by tables, clustered indexes, and nonclustered indexes.
- **sp_spaceused** computes the *rowtotal* value using the **rowcnt** built-in function. This function uses a value for the average number of rows per data page based on a value in the allocation pages for

the object. This method is very fast, but the results are estimates, and update and insert activity change actual values. The **update statistics** command, **dbcc checktable**, and **dbcc checkdb** update the rows-per-page estimate, so *rowtotal* is most accurate after running one of these commands. Always use **select count(*)** if you need exact row counts.

Messages

- `Object does not exist in this database.`

  The object specified does not exist in the current database.

- `Object is stored in 'sysprocedures' and has no space allocated directly.`

  The object is a trigger, stored procedure, rule, or default.

- `Object must be in the current database.`

  The object specified is not in the current database.

- `Views don’t have space allocated.`

  sp_spaceused reports only on the amount of space taken up by tables, clustered indexes, and nonclustered indexes.

Permissions

Any user can execute **sp_spaceused**.

Tables Used

#pgcounts, master.dbo.spt_values, master.dbo.sysusages, sysindexes, sysobjects

See Also

Commands	**create index, create table, drop index, drop table**
System procedures	**sp_help, sp_helpindex**

sp_syntax

Function

Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines (depending on which products and corresponding **sp_syntax** scripts exist on your server).

Syntax

```
sp_syntax {command | fragment} [, module_name]
   [, language]
```

Parameters

command – is the full name of a command or routine. To include spaces or Transact-SQL reserved words, enclosed the command in quotes.

fragment – is any fragment or portion of a command or routine name, such as “help” to list all system procedures providing help.

module – is the name or partial name of one of the modules, such as “Transact-SQL” or “Utility”. Each **sp_syntax** installation script adds different modules. Use **sp_syntax** without any parameters to see which modules exist on your server.

language – is the language of the syntax description to retrieve. *language* must be a valid language name in the *syslanguages* table.

Examples

1. **sp_syntax**

```
sp_syntax provides syntax help for Sybase products.
These modules are installed on this Server:

 Module
 --------------------
 OpenVMS
 Transact-SQL
 UNIX Utility
 System Procedure

Usage: sp_syntax command [, module [, language]]
```

Displays all **sp_syntax** modules available on your server.

2. **`sp_syntax "disk"`**

   Displays the syntax and functional description of all routines containing the word or word fragment “disk”. Since “disk” is a Transact-SQL reserved word, enclose it in quotes.

Comments

- The text for **sp_syntax** is in the database *sybsyntax*. Load **sp_syntax** and the *sybsyntax* database onto a server with the installation script described in the *System Administration Guide Supplement* for your platform. If you cannot access **sp_syntax**, see your System Administrator for information about installing it on your server.
- You can use wildcards within the command name you are searching for. If you are looking for commands or functions that contain the literal “_”, you may get unexpected results, since the wildcard “_” stands for “any single character”.

Messages

- `Can’t run sp_syntax from within a transaction.`

  sp_syntax creates temporary tables, so it cannot be run within a transaction.
- `No command or routine has a name like '%command%'`

  The command name you used is not in the *sybsyntax* database.
- `No module has a name like '%module%'`

  The module name you used is not in the *sybsyntax* database.
- `No command or routine has a name like '%command%' and a module like '%module%'`

  The combination of command name and module name is not in the *sybsyntax* database.

-
  ```
  sp_syntax provides syntax help for Sybase products.

  These modules are installed on this Server:
   Module
   --------------------
   module_name

  Usage: sp_syntax command [, module [, language]]
  ```

  These help message appear when you use **sp_syntax** with no arguments.

Permissions

Any user can execute **sp_syntax**.

Tables Used

sybsyntax..sybsyntax, *#tempsyntax1*, *#tempsyntax2*

sp_thresholdaction

Function

Executes automatically when the number of free pages on the log segment falls below the last-chance threshold (unless the threshold has been associated with a different procedure). **Sybase does not provide this procedure.**

Syntax

When a threshold is crossed, SQL Server passes the following parameters to the threshold procedure by position:

```
sp_thresholdaction @dbname,
       @segment_name,
       @space_left,
       @status
```

Parameters

@dbname – is the name of a database where the threshold was reached.

@segment_name – is the name of the segment where the threshold was reached.

@space_left – is the threshold size, in 2K pages.

@status – is 1 for the last-chance threshold; 0 for all other thresholds.

Examples

```
1. create procedure sp_thresholdaction
       @dbname varchar(30),
       @segmentname varchar(30),
       @space_left int,
       @status int
   as
       dump transaction @dbname to tapedump1
```

Creates a threshold procedure for the last-chance threshold that dumps the transaction log to a tape device.

Comments

- **sp_thresholdaction** must be created by the Database Owner (in a user database) or a System Administrator (in the *sybsystemprocs* database), or by a user with **create procedure** permission.

- You can add thresholds and create threshold procedures for any segment in a database.
- When the last-chance threshold is crossed, SQL Server searches for the **sp_thresholdaction** procedure in the database where the threshold event occurs. If it doesn't exist in that database, SQL Server searches for it in *sybsystemprocs*. If SQL Server does not find the procedure, it sends an error message to the error log.
- **sp_thresholdaction** should contain a **dump transaction** command to truncate the transaction log.
- By design, the last-chance threshold allows enough free space to record a **dump transaction** command. There may not be enough space to record additional user transactions against the database. Only commands that are not recorded in the transaction log (**select**, fast **bcp**, **readtext**, and **writetext**) and commands that might be necessary to free additional log space (**dump transaction**, **dump database**, and **alter database**) can be executed. By default, other commands are suspended and a message sent to the error log. To abort these commands rather than suspending them, use the "**abort tran on log full**" option of **sp_dboption** followed by the **checkpoint** command.

Waking Suspended Processes

- Once the **dump transaction** command frees sufficient log space, suspended process automatically awaken and complete.
- If fast **bcp**, **writetext**, or **select into** has resulted in unlogged changes to the database since the last backup, the last-chance threshold procedure cannot execute a **dump transaction** command. When this occurs, use **dump database** to make a copy of the database, then truncate the transaction log with **dump transaction**.
- If this does not free enough space to awaken the suspended processes, it may be necessary to increase the size of the transaction log. Use the **log on** option of the **alter database** command to allocate additional log space.
- As a last resort, System Administrators can use **sp_who** to determine which processes are suspended and the following command to awaken them:

```
select lct_admin("unsuspend", db_id)
```

See Also

Commands	**create procedure, dump transaction**
System procedures	**sp_addthreshold, sp_dboption, sp_dropthreshold, sp_helpsegment, sp_helpthreshold, sp_modifythreshold**

sp_unbindefault

Function

Unbinds a created default value from a column or from a user-defined datatype.

Syntax

```
sp_unbindefault objname [, futureonly]
```

Parameters

objname – is the name of either the table and column or the user-defined datatype from which to unbind the default. If the parameter is not of the form "*table.column*" then *objname* is taken to be a user-defined datatype. When unbinding a default from a user-defined datatype, any columns of that type that have the same default as the user-defined datatype had are also unbound. Columns of that type whose default has already been changed are unaffected.

futureonly – prevents existing columns of the specified user-defined datatype from losing their defaults.

Examples

1. **`sp_unbindefault "employees.startdate"`**

   Unbinds the default from the *startdate* column of the *employees* table.

2. **`sp_unbindefault ssn`**

   Unbinds the default from the user-defined datatype named *ssn*, and all columns of that type.

3. **`sp_unbindefault ssn, futureonly`**

   Unbinds defaults from the user-defined datatype *ssn*, but does not affect existing columns of type *ssn*.

Comments

- Use **sp_unbindefault** to remove defaults created with **sp_bindefault**. Use **alter table** to drop defaults declared using the **create table** or **alter table** statements.

- Columns of a user-defined datatype lose their current default unless their default had previously been changed, or the value of the optional second parameter is **futureonly**.
- To display the text of a default, execute **sp_helptext** with the default name as the parameter.

Messages

- `Column or usertype must be in current database.`

  The *objname* parameter cannot include a database reference.
- `Columns of the user datatype specified had their defaults unbound.`

  Defaults on other columns of the user-defined datatype specified were unbound, unless their defaults were changed previously.
- `Default unbound from datatype.`

  The user-defined datatype supplied for the *objname* parameter no longer has any default.
- `Default unbound from table column.`

  The table column supplied for the *objname* parameter no longer has any default.
- `The specified column has no default.`

  No default is bound to the column name supplied for the *objname* parameter.
- `The specified user datatype has no default.`

  No default is bound to the datatype name supplied for the *objname* parameter.
- `You do not own a table with a column of that name.`

  The table name supplied for the *objname* parameter either doesn't exist in the database or you don't own it. You can only bind or unbind defaults from columns in tables that you own.
- `You do not own a user datatype of that name.`

  The user-defined datatype supplied for the *objname* parameter either doesn't exist in the database or you don't own it. You can only bind or unbind defaults from datatypes that you own.

Permissions

Only the object owner can execute **sp_unbindefault**.

Tables Used

syscolumns, sysobjects, sysprocedures, systypes

See Also

Commands	**create default, drop default**
System procedures	**sp_bindefault, sp_helptext**

sp_unbindmsg

Function

Unbinds a user-defined message from a constraint.

Syntax

```
sp_unbindmsg constraint_name
```

Parameters

constraint_name – is the name of the constraint from which you are unbinding a message.

Examples

1. **sp_unbindmsg positive_balance**

   Unbinds a user-defined message from the constraint *positive_balance*.

Comments

- You can bind only one message to a constraint. To change the message bound to a constraint, use **sp_bindmsg**; the new message number replaces any existing bound message. It is not necessary to use **sp_unbindmsg** first.
- To retrieve message text from the *sysusermessages* table, execute **sp_getmessage**.

Messages

- `Constraint name must be in 'current' database.`

  You can only unbind messages from constraints that are defined in the current database.

- `Constraint name must belong to the current user.`

  You cannot unbind a message from a constraint created by another user.

- `No such referential or check constraint exists. Please check whether the constraint name is correct.`

  Use **sp_help** *tablename* to see a list of all existing constraints on a table.

- `Constraint is not bound to any message.`

  No message is currently bound to *constraint_name*.
- `Unbinding message failed unexpectedly. Please try again.`

  An error occurred. Reissue the command.
- `Message unbound from constraint.`

  You have successfully unbound the user-defined message from *constraint_name*.

Permissions

Only the object owner can execute **sp_unbindmsg**.

Tables Used

sysconstraints, sysobjects

See Also

System procedures	**sp_addmessage, sp_bindmessage, sp_getmessage**

sp_unbindrule

Function

Unbinds a rule from a column or from a user-defined datatype.

Syntax

sp_unbindrule *objname* [, futureonly]

Parameters

objname – is the name of the table and column or of the user-defined datatype from which the rule is to be unbound. If the parameter is not of the form "*table.column*", then *objname* is taken to be a user-defined datatype. Unbinding a rule from a user-defined datatype also unbinds it from columns of that type. This has no effect on columns that are already bound to a different rule.

futureonly – prevents existing columns of the specified user-defined datatype from losing their rules.

Examples

1. **sp_unbindrule "employees.startdate"**

   Unbinds the rule from the *startdate* column of the *employees* table.

2. **sp_unbindrule def_ssn**

   Unbinds the rule from the user-defined datatype named *def_ssn* and all columns of that type.

3. **sp_unbindrule ssn, futureonly**

   The user-defined datatype *ssn* no longer has a rule, but no existing *ssn* columns are affected.

Comments

- Executing **sp_unbindrule** removes a rule from a column or from a user-defined datatype in the current database. If you don't want to unbind the rule from existing *objname* columns, use the string **futureonly** as the second parameter.
- You cannot use **sp_unbindrule** to unbind a check constraint. Use **alter table** to drop the constraint.
- To unbind a rule from a table column, specify the *objname* argument in the format "*table.column.*"

- The rule is unbound from all existing columns of the user-defined datatype unless their rule had previously been changed, or you specify **futureonly.**
- To display the text of a rule, execute **sp_helptext** with the rule name as the parameter.

Messages

- `Column or usertype must be in current database.`

  The *objname* parameter may not include a database reference.
- `Columns of the user datatype specified had their rules unbound.`

  Rules on other columns of the user-defined datatype specified were unbound, unless their rules had previously been changed.
- `Rule unbound from datatype.`

  The user-defined datatype supplied for the *objname* parameter no longer has any rule.
- `Rule unbound from table column.`

  The table column supplied for the *objname* parameter no longer has any rule.
- `The specified column has no rule.`

  There is no rule bound to the table column supplied for the *objname* parameter. Nothing changed.
- `The specified user datatype has no rule.`

  There is no rule bound to the user-defined datatype supplied for the *objname* parameter. Nothing changed.
- `You do not own a table with a column of that name.`

  The table name supplied for the *objname* parameter either doesn't exist in the database or you don't own it. You can only bind or unbind rules on tables that you own.
- `You do not own a user datatype of that name.`

  The user-defined datatype supplied for the *objname* parameter either doesn't exist in the database or you don't own it. You can only bind or unbind rules from datatypes that you own.

Permissions

Only the object owner can execute **sp_unbindrule**.

Tables Used

syscolumns, sysconstraints, sysobjects, sysprocedures, systypes

See Also

Commands	**create rule, drop rule**
System procedures	**sp_bindrule, sp_helptext**

sp_volchanged

Function

Notifies the Backup Server that the operator performed the requested volume handling during a dump or load.

Syntax

```
sp_volchanged session_id, device_name, action
   [, filename [, volume_name]]
```

Parameters

session_id – identifies the Backup Server session that requested the volume change. Use the *@session_id* parameter specified in the Backup Server's volume change request.

device_name – is the device on which a new volume was mounted. Use the *@devname*e parameter specified in the Backup Server's volume change request. If the Backup Server is not located on the same machine as the SQL Server, use the form:

```
device at backup_server_name
```

action – indicates whether the Backup Server should **abort**, **proceed** with, or **retry** the dump or load.

filename – is the file to load. If you do not specify a file name with **sp_volchanged**, the Backup Server loads the **file** = *filename* parameter of the load command. If neither **sp_volchanged** nor the load command specifies which file to load, the Backup Server loads the first file on the tape.

volume_name – is the volume name that appears in the ANSI tape label. The Backup Server writes the volume name in the ANSI tape label when overwriting an existing dump, dumping to a brand new tape, or dumping to a tape whose contents are not recognizable. If you do not specify a *volume_name* with **sp_volchanged**, the Backup Server uses the **dumpvolume** value specified in the dump command. If neither **sp_volchanged** nor the dump command specifies a volume name, the Backup Server leaves the name field of the ANSI tape label blank.

During loads, the Backup Server uses the *volume_name* to confirm that the correct tape has been mounted. If you do not specify a *volume_name* with **sp_volchanged**, the Backup Server uses the **dumpvolume** specified in the load command. If neither

sp_volchanged nor the load command specifies a volume name, the Backup Server does not check the name field of the ANSI tape label before loading the dump.

Examples

1. **sp_volchanged 8, "/dev/nrmt4", RETRY**

   This message from Backup Server indicates that a mounted tape's expiration date has not been reached:

```
Backup Server: 4.49.1.1: OPERATOR: Volume to be overwritten on
'/dev/rmt4' has not expired: creation date on this volume is
Sunday, Nov. 15, 1992, expiration date is Wednesday, Nov. 25,
1992.
Backup Server: 4.78.1.1: EXECUTE sp_volchanged
        @session_id = 8,
        @devname = '/auto/remote/pubs3/SERV/Masters/testdump',
        @action = { 'PROCEED' | 'RETRY' | 'ABORT' }
```

   The operator changes the tape, then issues the command in Example 1.

Comments

Roles of Operator, SQL Server, and Backup Server in Volume Changes

- If the Backup Server detects a problem with the currently mounted volume, it requests a volume change:
  - On OpenVMS systems, the Backup Server sends volume change messages to the operator terminal on the machine on which it is running. Use the **with notify = client** option of the dump or load command to route other Backup Server messages to the terminal session on which the **dump** or **load** request initiated.
  - On UNIX systems, the Backup Server sends messages to the client that initiated the dump or load request. Use th**e with notify = operator_console** option of the dump or load command to route messages to the terminal where the Backup Server was started.
- After mounting another volume, the operator executes **sp_volchanged** from any SQL Server that can communicate with the Backup Server performing the dump or load. The operator does not have to log into the SQL Server on which the dump or load originated.
- On OpenVMS systems, the operating system—not the Backup Server—requests a volume change when it detects the end of a

volume or when the specified drive is offline. The operator uses the OpenVMS REPLY command to reply to these messages.

- On UNIX systems, the Backup Server requests a volume change when the tape capacity has been reached. The operator mounts another tape, then executes **sp_volchanged**. *Figure 1-1: Changing Tape Volumes on a UNIX System* illustrates this process.

Time	Operator, using isql	SQL Server	Backup Server
	Issues the **dump database** command		
		Sends dump request to Backup Server	
			Receives dump request message from SQL Server Sends message for tape mounting to operator Waits for operator's reply
	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		
			Checks tapes If tapes are okay, begins dump When tape is full, sends volume change request to operator
	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		
			Continues dump When dump is complete, sends messages to operator and SQL Server
	Receives message that dump is complete Removes and labels tapes	Receives message that dump is complete Releases locks Completes the **dump database** command	

Figure 1-1: Changing Tape Volumes on a UNIX System

Messages

Volume Change Prompts for Loads

- `Dumpfile '`*`file_name`*`' section `*`volume_name`*` found instead of '`*`file_name`*`' section `*`volume_name`*`.`

  The Backup Server issues this message if it cannot find the specified file on a single-file medium.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and try to load it	**sp_volchanged** *session_id*, *device_name*, **retry** [, *file_name* [, *volume_name*]]
Load the file on the currently mounted volume, even though it is not the specified file (not recommended)	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Mount the next volume to read.`

  The Backup Server issues this message when it is ready to read the next section of the dump file from a multi-volume dump.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount the next volume and proceed with the load	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Mount the next volume to search.`

  The Backup Server issues this message if it cannot find the specified file on multi-file medium.

The operator can	By replying
Abort the load	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and proceed with the load	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

Volume Change Prompts for Dumps

- `Mount the next volume to search.`

  When appending a dump to an existing volume, the Backup Server issues this message if it cannot find the end-of-file mark.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount a new volume and proceed with the dump	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Mount the next volume to write.`

  The Backup Server issues this message when it reaches the end of the tape. This occurs when it detects the end-of-tape mark, or dumps the number of kilobytes specified by the **capacity** parameter of the dump command, or the device's *high* value from the *sysdevices* system table.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount the next volume and proceed with the dump	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Volume on device` *`device_name`* `has restricted access (code` *`access_code`*`).`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a tape with ANSI access restrictions without specifying the **init** option.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *device_name*, **retry** [, *file_name* [, *volume_name*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Volume on device` *`device_name`* `is expired and will be overwritten.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. During dumps to single-file media, Backup Server issues this message if you have not specified the init option and the tape contains a dump whose expiration date has passed.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *device_name*, **retry** [, *file_name* [, *volume_name*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Volume to be overwritten on '`*`device_name`*`' has not expired: creation date on this volume is` *`creation_date`*`, expiration date is` *`expiration_date`*`.`

  On single-file media, the Backup Server checks the expiration date of any existing dump unless you specify the **init** option. The Backup Server issues this message if the dump has not yet expired.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *device_name*, **retry** [, *file_name* [, *volume_name*]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

- `Volume to be overwritten on '`*`device_name`*`' has unrecognized label data.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a new tape or a tape with non-Sybase data without specifying the **init** option.

The operator can	By replying
Abort the dump	**sp_volchanged** *session_id*, *device_name*, **abort**
Mount another volume and retry the dump	**sp_volchanged** *session_id*, *device_name*, **retry** [, *file_name* [, *volume_name*]]
Proceed with the dump, overwriting any existing contents of the volume	**sp_volchanged** *session_id*, *device_name*, **proceed** [, *file_name* [, *volume_name*]]

Permissions

Any user can execute **sp_volchanged** to respond to a volume change request. This need not be the same user who started the dump or load.

Tables Used

master..sysdevices, sysobjects

See Also

Commands	**dump database, dump transaction, load database, load transaction**
Topics	Roles

sp_who

Function

Reports information about all current SQL Server users and processes, or about a particular user or process.

Syntax

sp_who [*login_name* | "*spid*"]

Parameters

login_name – is the SQL Server login name of a user to report on.

spid – is the number of a specific process to report on. Enclose process numbers in quotes (SQL Server expects a *char* type).

Examples

1. **sp_who**

   Reports on the processes running on SQL Server:

```
spid  status     loginame hostname blk dbname cmd
 ---- --------    -------- -------- --- ------ -------------
  1  recv sleep  bird     jazzy    0   master AWAITING COMMAND
  2  sleeping    NULL              0   master NETWORK HANDLER
  3  sleeping    NULL              0   master MIRROR HANDLER
  4  sleeping    NULL              0   master AUDIT PROCESS
  5  sleeping    NULL              0   master CHECKPOINT SLEEP
  6  recv sleep  rose     petal    0   master AWAITING COMMAND
  7  running     sa       helos    0   master SELECT
  8  send sleep  daisy    chain    0   pubs2  SELECT
  9  alarm sleep lily     pond     0   master WAITFOR
 10  lock sleep  viola    cello    7   pubs2  SELECT
```

   The *spid* column contains the process identification numbers that are used in the Transact-SQL **kill** command. The *blk* column contains the process ID's of the blocking process, if there is one. A blocking process (which may be infected or have an exclusive lock) is one that is holding resources that another process needs. In the previous example, process 10 (a **select** on a table) is blocked by process 7 (a **begin transaction** followed by an **insert** on the same table).

   If you enable mirrored disks or remote procedure calls, the mirror handler and the site handler also appear in the report from **sp_who**.

2. **`sp_who victoria`**

   Reports on the processes the user "victoria" is running.

3. **`sp_who "17"`**

   Reports what SQL Server process number 17 is doing.

Comments

- **sp_who** reports information about a specified user or SQL Server process. Without parameters, **sp_who** reports which users are running what processes in all databases.
- Running **sp_who** on a single-engine server shows the **sp_who** process "running" and all other processes "runnable" or in one of the sleep states. In multi-engine servers, there can be a "running" process for each engine.
- **sp_who** reports NULL in the *loginame* column for all system processes.
- System Administrators can remove many processes with the **kill** command. See **kill** in the *SQL Server Reference Manual, Volume 1*.

Messages

- `No login with the specified name exists.`

  The name supplied for the *login_name* parameter does not exist on SQL Server.

Permissions

Any user can execute **sp_who**.

Tables Used

master..sysprocesses

See Also

Commands	**kill**
System procedures	**sp_lock**

Catalog Stored Procedures

2 Catalog Stored Procedures

This chapter describes the catalog stored procedures. Catalog stored procedures retrieve information from the system tables in tabular form.

The following table lists the catalog stored procedures that are covered in this chapter.

Procedure	Description
sp_column_privileges	Returns permissions information for one or more columns in a table or view.
sp_columns	Returns information about the type of data that can be stored in one or more columns.
sp_databases	Returns a list of databases on a server.
sp_datatype_info	Returns information about a particular datatype or about all supported datatypes.
sp_fkeys	Returns logical foreign key information for the current database. Foreign keys must have been declared through the ANSI integrity constraint mechanism.
sp_pkeys	Returns primary key information for a single table. Primary keys must have been declared through the ANSI integrity constraint mechanism.
sp_server_info	Returns a list of attribute names and matching values for a server.
sp_special_columns	Returns the optimal set of columns that uniquely identify a row in a table or view; can also return a list of the columns that are automatically updated when any value in the row is updated by a transaction.
sp_sproc_columns	Returns information about a stored procedure’s input and return parameters.
sp_statistics	Returns a list of indexes on a single table.
sp_stored_procedures	Returns information about one or more stored procedures.
sp_table_privileges	Returns privilege information for all columns in a table or view.
sp_tables	Returns a list of objects that can appear in a **from** clause.

Table 2-1: Catalog Stored Procedures

Syntax and Optional Parameters

In many cases it is more convenient to supply parameters to the catalog stored procedures in the form:

```
@parametername = value
```

than to supply all of the parameters. The parameter names in the syntax statements match the parameter names defined by the procedures.

For example, the syntax for **sp_columns** is:

```
sp_columns table_name [, table_owner]
[, table_qualifier] [, column_name]
```

To use **sp_columns** to find information about a particular column, you can use:

```
sp_columns publishers, @column_name = "pub_id"
```

This provides the same information as the command with all of the parameters specified:

```
sp_columns publishers, "dbo", "pubs2", "pub_id"
```

You can also use “null” as a placeholder:

```
sp_columns publishers, null, null, "pub_id"
```

System Procedure Tables

The catalog stored procedures **sp_columns**, **sp_datatype_info**, **sp_special_columns**, and **sp_sproc_columns** use the catalog stored procedure tables *spt_datatype_info*, *spt_datatype_info_ext*, and *spt_server_info* the *sybsystemprocs* database to convert internal system values (for example, status bits) into human-readable format.

In addition, **sp_column_privileges** and **sp_table_privileges** create and then drop the temporary tables *#column_privileges*, *#distinct_grantors*, *#results_table*, *#sysprotects*, *#useful_groups*.

Pattern Matching

SQL Server offers a wide range of pattern matching through regular expressions. However, for maximum interoperability, assume only ANSI SQL pattern matching (the **%** and _ wildcards).

ODBC Datatypes

Table 2-2 and Table 2-3 list the datatype code numbers and matching datatype names that **sp_columns** and **sp_sproc_columns** return in the DATA_TYPE column. The source for the description is the Open Database Connectivity API.

Datatypes

Name	Type
char	1
decimal	3
double precision	8
float	6
integer	4
numeric	2
real	7
smallint	5
varchar	12

Table 2-2: Datatypes

Extended Datatypes

Name	Type
bigint	-5
binary (bit datatype)	-2
bit	-7
date	9
long varbinary	-4
long varchar	-1
time	10
timestamp	11
tinyint	-6
varbinary (bit varying datatype)	-3

Table 2-3: Extended Datatypes

sp_column_privileges

Function

Returns permissions information for one or more columns in a table or view.

Syntax

```
sp_column_privileges table_name [, table_owner
   [, table_qualifier [, column_name]]]
```

Parameters

table_name – is the name of the table. No wildcard pattern matching is supported.

table_owner – is the name of the table owner. No wildcard pattern matching is supported. If you do not specify the table's owner, **sp_column_privileges** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. Acceptable values are the name of the current database and NULL.

column_name – is the name of the column whose permissions you want to display. Use wildcards to request information for more than one column. If you do not specify a column name, **sp_column_privileges** returns permissions information for all columns in the specified table.

Examples

1. **sp_column_privileges discounts, null, null, discounttype**

```
table_qualifier table_owner
     table_name column_name
     grantor grantee
     privilege is_grantable
---------------------- -------------------------
    ---------------------- ---------------------
    ---------------------- ----------------------
    ------------------------------ -----------
pubs2 dbo
 discounts discounttype
 dbo guest
 INSERT NO
```

```
pubs2 dbo
 discounts discounttype
 dbo guest
 SELECT NO
pubs2 dbo
 discounts discounttype
 dbo guest
 UPDATE NO
pubs2 dbo
 discounts discounttype
 dbo guest
 REFERENCE NO
pubs2 dbo
 discounts discounttype
 dbo dbo
 INSERT YES
pubs2 dbo
 discounts discounttype
 dbo dbo
 SELECT YES
pubs2 dbo
 discounts discounttype
 dbo dbo
 UPDATE YES
pubs2 dbo
 discounts discounttype
 dbo dbo
 REFERENCE YES
```

Comments

- The following table describes the results set:

Column	Datatype	Description
table_qualifier	*varchar(32)*	The database name. This field can be NULL.
table_owner	*varchar(32)*	
table_name	*varchar(32)*	NOT NULL
column_name	*varchar(32)*	
grantor	*varchar(32)*	NOT NULL
grantee	*varchar(32)*	NOT NULL

Table 2-4: Results Set for sp_column_privileges

Column	Datatype	Description
privilege	*varchar(32)*	Identifies the column privilege. May be one of the following: SELECT - The grantee is permitted to retrieve data for the column. INSERT - The grantee is permitted to provide data for the column in new rows that are inserted into the associated table. UPDATE - The grantee is permitted to update data in the column. REFERENCE - The grantee is permitted to refer to the column within a constraint (for example, a unique, referential, or table check constraint).
is_grantable	*varchar(3)*	Indicates whether the grantee is permitted to grant the privilege to other users. The values are YES, NO, or NULL.

Table 2-4: Results Set for sp_column_privileges (continued)

Messages

- `Catalog procedure sp_column_privileges can not be run in a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.

- `Object name must be qualified with the owner name.`
- `Object name can only be qualified with owner name.`
- `This may be a temporary object. Please execute procedure from tempdb.`

  You invoked **sp_column_priviliges** for a table beginning with "#". Execute the **use** command to switch to *tempdb*, then rerun **sp_column_privileges**.

- `The table or view named doesn’t exist in the current database.`

  The specified table or view does not exist. Check the spelling of the *table_name*.

- `The table does not have a column named` *`column_name`*`.`

  The specified column does not belong to the table.

- `Table qualifier must be name of current database.`

  sp_column_privileges cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_column_privileges**.

Permissions

Any user can execute **sp_column_privileges**.

Tables Used

syscolumns, sysobjects, sysusers

sp_columns

Function

Returns information about the type of data that can be stored in one or more columns.

Syntax

```
sp_columns table_name [, table_owner ]
   [, table_qualifier] [, column_name]
```

Parameters

table_name – is the name of the table or view. Use wildcards to request information about more than one table.

table_owner – is the owner of the table or view. Use wildcards to request information about tables owned by more than one user. If you do not specify a table owner, **sp_columns** looks first for tables owned by the current user and then for tables owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

column_name – is the name of the column for which you want information. Use wildcards to request information about more than one column.

Examples

1. **`sp_columns "publishers", null, null, "p%"`**

   Displays information about all columns in the *publishers* table that begin with “p”.

2. **`sp_columns "s%", null, null, "st%"`**

   Displays information about all columns beginning with “st” in tables that begin with “s”.

Comments

- The following table shows the results set:

Column	Datatype	Description
table_qualifier	*varchar(32)*	The database name. This field can be NULL.
table_owner	*varchar(32)*	
table_name	*varchar(32)*	NOT NULL.
column_name	*varchar(32)*	NOT NULL.
data_type	*smallint*	Integer code for ODBC datatype. If this is a datatype that cannot be mapped into an ODBC type, it is NULL.
type_name	*varchar(30)*	String representing a datatype. The underlying DBMS presents this datatype name.
precision	*int*	Number of significant digits.
length	*int*	Length in bytes of a datatype.
scale	*smallint*	Number of digits to the right of the decimal point.
radix	*smallint*	Base for numeric types.
nullable	*smallint*	The value 1 means NULL is possible; 0 means NOT NULL.
remarks	*varchar(254)*	
ss_data_type	*smallint*	A SQL Server datatype.
colid	*tinyint*	A column appended to the result set.

Table 2-5: Results Set for sp_columns

Messages

- `Table qualifier must be name of current database.`

  sp_columns cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_columns**.

Permissions

Any user can execute **sp_columns**.

Tables Used

syscolumns, sysobjects, systypes, sybsystemprocs..spt_datatype_info

sp_databases

Function

Returns a list of databases on a SQL Server.

Syntax

`sp_databases`

Parameters

None.

Examples

1. **`sp_databases`**

```
database_name                          database_size
         remarks

 ------------------------------ -------------
        -----------------------------------------

 master                                      3072
         NULL

 model                                       2048
         NULL

 mydb                                        2048
         NULL

 pubs2                                       2048
         NULL

 sybsecurity                                 5120
         NULL

 sybsystemprocs                             10240
         NULL

 tempdb                                      2048
         NULL
```

Comments

- Table 2-6 shows the result set:

Column	Datatype	Description
database_name	*char*(32)	NOT NULL database name.
database_size	*int*	Size of database in kilobytes.
remarks	*varchar*(254)	SQL Server always returns NULL.

Table 2-6: Results Set for sp_databases

Permissions

Any user can execute **sp_databases**.

Tables Used

#databases, master..sysdatabases, master..sysusages, sysobjects

sp_datatype_info

Function

Returns information about a particular datatype or about all supported datatypes.

Syntax

sp_datatype_info [*data_type*]

Parameters

data_type – is the code number for the specified datatype about which information is returned. Datatype codes are listed in *Table 2-2: Datatypes* and *Table 2-3: Extended Datatypes*.

Comments

- The following table describes the results set:

Column	Datatype	Description
type_name	*varchar(30)*	A DBMS-dependent datatype name (the same as the *type name* column in the **sp_columns** results set).
data_type	*smallint*	A code for the ODBC type to which all columns of this type are mapped.
precision	*int*	The maximum precision for the datatype on the data source. Zero is returned for datatypes where precision is not applicable.
literal_prefix	*varchar(32)*	Character(s) used to prefix a literal. For example, a single quotation mark (') for character types and 0x for binary.
literal_suffix	*varchar(32)*	Character(s) used to terminate a literal. For example, a single quotation mark (') for character types and nothing for binary.
create_params	*varchar(32)*	A description of the creation parameters for this datatype.
nullable	*smallint*	The value 1 means this datatype can be created allowing null values; 0 means it cannot.

Table 2-7: Results Set for sp_datatype_info

Column	Datatype	Description
case_sensitive	*smallint*	The value 1 means all columns of this type are case sensitive (for collations); 0 means they are not.
searchable	*smallint*	The value 1 means columns of this type can be used in a **where** clause.
unsigned_attribute	*smallint*	The value 1 means the datatype is unsigned; 0 means the datatype is signed.
money	*smallint*	The value 1 means it is a money datatype; 0 means it is not.
auto_increment	*smallint*	The value 1 means the datatype is automatically incremented; 0 means it is not.
local_type_name	*varchar*(128)	Localized version of the data source dependent name of the datatype.

Table 2-7: Results Set for sp_datatype_info (continued)

Permissions

Any user can execute **sp_datatype_info**.

Tables Used

sybsystemprocs..spt_datatype_info, systypes, sysdatabases, sysmessages, sysprocesses

sp_fkeys

Function

Returns logical foreign key information for the current database. Foreign keys must have been declared through the ANSI integrity constraint mechanism.

Syntax

```
sp_fkeys pktable_name [, pktable_owner]
   [, pktable_qualifier] [, fktable_name]
   [, fktable_owner] [, fktable_qualifier]
```

Parameters

pktable_name – is the name of the primary key table. No wildcard pattern matching is supported. You must specify either this parameter or the *fktable_name* parameter, or both.

pktable_owner – is the name of the primary key table owner. No wildcard pattern matching is supported. If you do not specify the table owner, **sp_fkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

pktable_qualifier – is the name of the database that contains the primary key table. This can be either the current database or NULL.

fktable_name – is the name of the foreign key table. No wildcard pattern matching is supported. Either this parameter or the *pktable_name* parameter, or both, must be given.

fktable_owner – is the name of the foreign key table owner. No wildcard pattern matching is supported. If the parameter is not specified, **sp_fkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

fktable_qualifier – is the name of the database that contains the foreign key table. This can be either the current database or NULL.

Comments

- Table 2-8 describes the results set:

Column	**Datatype**	**Description**
pktable_qualifier	*varchar*(32)	The database that contains the primary key table.
pktable_owner	*varchar(32)*	The owner of the primary key table.
pktable_name	*varchar(32)*	NOT NULL.
pkcolumn_name	*varchar(32)*	NOT NULL.
fktable_qualifier	*varchar(32)*	The database that contains the foreign key table.
fktable_owner	*varchar(32)*	The owner of the foreign key table.
fktable_name	*varchar(32)*	NOT NULL.
fkcolumn_name	*varchar(32)*	NOT NULL.
key_seq	*smallint*	NOT NULL. The sequence number of the column in a multi-column primary key.
update_rule	*smallint*	Action to be applied to the foreign key when the SQL operation is UPDATE. Zero is returned for this column.
delete_rule	*smallint*	Action to be applied to the foreign key when the SQL operation is DELETE. Zero is returned for this column.

Table 2-8: Results Set for sp_fkeys

- Both the primary key and foreign key must have been declared through the ANSI integrity constraint mechanism. See **create table** in the *SQL Server Reference Manual,* Volume 1 for more information.
- If the primary key table name is supplied but the foreign key table name is NULL, this procedure returns all tables that include a foreign key to the given table. If the foreign key table name is supplied, but the primary key table name is NULL, the procedure returns all tables related by a primary key / foreign key relationship to foreign keys in the foreign key table.
- This procedure does not support the *common key* type as specified in the SQL Server *syskeys* catalog.

Messages

- `Catalog procedure sp_fkeys can not be run in a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.

- `Foreign key table qualifier must be name of current database.`

  sp_fkeys cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_fkeys**.

- `Primary key table qualifier must be name of current database.`

  sp_fkeys cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_fkeys**.

- `Object does not exist in this database.`

  The primary key table or foreign key table does not exist in the current database as specified. Check the spelling of the table name.

- `Primary key table name or foreign key table name or both must be given.`

  You must specify the name of the primary key table, the foreign key table, or both.

Permissions

Any user can execute **sp_fkeys**.

Tables Used

#fid, #fkey_res, #fkeys, #pid, #pkeys, sysobjects, sysreferences

See Also

Commands	**create table**

sp_pkeys

Function

Returns primary key information for a single table. Primary keys must have been declared through the ANSI integrity constraint mechanism.

Syntax

```
sp_pkeys table_name [, table_owner]
   [, table_qualifier]
```

Parameters

table_name – is the name of the table. No wildcard pattern matching is supported.

table_owner – is the name of the table owner. No wildcard pattern matching is supported. If the parameter is not specified, **sp_pkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database that contains the table. This can be either the current database or NULL.

Comments

- Table 2-9 shows the results set:

Column	Datatype	Description
table_qualifier	*varchar*(32)	The database name. This field can be NULL.
table_owner	*varchar*(32)	
table_name	*varchar*(32)	NOT NULL.
column_name	*varchar*(32)	NOT NULL.
key_seq	*smallint*	NOT NULL. The sequence number of the column in a multi-column primary key.

Table 2-9: Results Set for sp_pkeys

- Primary keys must have been declared through the ANSI integrity constraint mechanism in the **create table** statement.
- The term *primary key* refers to a logical primary key for a table. SQL Server expects that every logical primary key has a unique

index defined on it, and that this unique index is also returned in **sp_statistics**.

Messages

- `Object does not exist in this database.`

  The primary key table or foreign key table does not exist in the current database as specified. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_pkeys cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_pkeys**.

- `Catalog procedure sp_pkeys can not be run in a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.

Permissions

Any user can execute **sp_pkeys**.

Tables Used

#pkeys, sysindexes, sysobjects

See Also

Commands	**create table**

sp_server_info

Function

Returns a list of attribute names and matching values for SQL Server.

Syntax

sp_server_info [*attribute_id*]

Parameters

attribute_id – is the integer ID of the attribute.

Examples

1. **sp_server_info 109**

```
attribute_id attribute_name
         attribute_value


 ------------ ------------------------------------
        -------------------------------------------
           12 MAX_OWNER_NAME_LENGTH
         30
```

2. **sp_server_info**

   Returns the list attributes described by the mandatory rows, and their values.

Comments

- Table 2-10 describes the results set:

Column	Datatype	Description
attribute_id	*int*	NOT NULL.
attribute_name	*varchar*(60)	NOT NULL.
attribute_value	*varchar*(255)	

Table 2-10: Results Set for sp_server_info

- Table 2-11 shows the mandatory rows in this results set:

ID	Name	Description	Value
1	DBMS_NAME	Name of the DBMS.	SQL SERVER
2	DBMS_VER	Version of the DBMS.	*@@version*
6	DBE_NAME		
10	OWNER_TERM	SQL Server’s term for a table owner (the second part of a three-part name).	owner
11	TABLE_TERM	SQL Server's term for a table (the third part of a three-part name).	table
12	MAX_OWNER_NAME_LENGTH	Maximum length of the name for a table owner (the second part of a three-part name).	30
16	IDENTIFIER_CASE	The case sensitivity of user-defined names (table names, column names, stored procedure names) in the database (the case in which these objects are presented in the system catalogs).	MIXED
15	COLUMN_LENGTH	The maximum number of characters for a column name.	30
13	TABLE_LENGTH	The maximum number of characters for a table name.	30
100	USERID_LENGTH	The maximum number of characters for a user name.	30
17	TX_ISOLATION	The initial transaction isolation level the server assumes, corresponding to an isolation level defined in ANSI SQL2.	2
18	COLLATION_SEQ	The assumed ordering of the character set for this server.	
14	MAX_QUAL_LENGTH	Maximum length of the name for a table qualifier (the first part of a three-part table name).	30
101	QUALIFIER_TERM	SQL Server's term for a table qualifier (the first part of a three-part name).	database
19	SAVEPOINT_SUPPORT	Does the underlying DBMS support named savepoints?	Y

Table 2-11: Mandatory Results Returned by sp_server_info

ID	Name	Description	Value
20	MULTI_RESULT_SETS	Does the underlying DBMS or the gateway itself support multiple results sets (can multiple statements be sent through the gateway, with multiple results sets returned to the client)?	Y
102	NAMED_TRANSACTIONS	Does the underlying DBMS support named transactions?	Y
103	SPROC_AS_LANGUAGE	Can stored procedures be executed as language events?	Y
103	REMOTE_SPROC	Can stored procedures be executed through the remote stored procedure APIs in DB-Library?	Y
22	ACCESSIBLE_TABLES	In the **sp_tables** stored procedure, does the gateway return only tables, views, and so on, that are accessible by the current user (that is, the user who has at least **select** privileges for the table)?	Y
104	ACCESSIBLE_SPROC	In the **sp_stored_procedures** stored procedure, does the gateway return only stored procedures that are executable by the current user?	Y
105	MAX_INDEX_COLS	Maximum number of columns in an index for the DBMS.	16
106	RENAME_TABLE	Can tables be renamed?	Y
107	RENAME_COLUMN	Can columns be renamed?	Y
108	DROP_COLUMN	Can columns be dropped?	Y
109	INCREASE_COLUMN_LENGTH	Can column size be increased?	N
110	DDL_IN_TRANSACTION	Can DDL statements appear in transactions?	Y
111	DESCENDING_INDEXES	Are descending indexes supported?	N
112	SP_RENAME	Can a stored procedure be renamed?	Y
500	SYS_SPROC_VERSION	The version of the catalog stored procedures currently implemented.	01.01.2822

Table 2-11: Mandatory Results Returned by sp_server_info (continued)

Messages

- `Attribute id` *`attribute_id`* `is not supported.`

  Check the spelling of the attribute.

Permissions

Any user can execute **sp_server_info**.

Tables Used

sybsystemprocs..spt_server_info, sysobjects

sp_special_columns

Function

Returns the optimal set of columns that uniquely identify a row in a table or view; can also return a list of the columns that are automatically updated when any value in the row is updated by a transaction.

Syntax

```
sp_special_columns table_name [, table_owner]
   [, table_qualifier] [, col_type]
```

Parameters

table_name – is the name of the table or view. No wildcard pattern matching is supported.

table_owner – is the name of the table or view owner. No wildcard pattern matching is supported. If you do not specify the table owner, **sp_special_columns** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

col_type – is "R" to return information about columns whose values uniquely identify any row in the table, or "V" to return information about columns whose values are automatically generated by SQL Server each time a row is inserted or updated.

Examples

1. **sp_special_columns systypes**

```
scope  column_name                  data_type type_name     precision
         length      scale
 ------ ----------------------- --------- ------------ --------
         ----------- ------
      0 name                           12 varchar               30
                  30   NULL
```

Comments

- Table 2-12 describes the results set:

Column	Datatype	Description
scope	*int*	NOT NULL. Actual scope of the row id. SQL Server always returns 0.
column_name	*varchar(30)*	NOT NULL. Column identifier.
data_type	*smallint*	The integer code for an ODBC datatype. If this datatype cannot be mapped into an ANSI type, the value will be NULL. The native datatype name is returned in the *type_name* column. (See the ODBC datatypes table at the beginning of this chapter.)
type_name	*varchar(13)*	The string representation of the datatype. This is the datatype name as presented by the underlying DBMS.
precision	*int*	The number of significant digits.
length	*int*	The length in bytes of the datatype.
scale	*smallint*	The number of digits to the right of the decimal point.

Table 2-12: Results Set for sp_special_columns

Messages

- `There is no table named` *`table_name`* `in the current database.`

  The table does not exist in the current database as specified. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_special_columns cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_special_columns**.

- `Illegal value for 'col_type' argument. Legal values are 'V' or 'R'.`

  You must specify V or R.

Permissions

Any user can execute **sp_special_columns**.

Tables Used

sybsystemprocs..spt_datatype_info, syscolumns, sysindexes, sysobjects, systypes, sysusers

sp_sproc_columns

Function

Returns information about a stored procedure's input and return parameters.

Syntax

sp_sproc_columns *sp_name* [, *sp_owner*]
[, *sp_qualifier*] [, *column_name*]

Parameters

sp_name – is the name of the stored procedure. No wildcard pattern matching is supported.

sp_owner – is the owner of the stored procedure. No wildcard pattern matching is supported. If you do not specify the owner of the procedure, **sp_sproc_columns** looks first for a procedure owned by the current user and then for a procedure owned by the Database Owner.

sp_qualifier – is the name of the database. This can be either the current database or NULL.

column_name – is the name of the parameter about which you want information. If you do not supply a parameter name, **sp_sproc_columns** returns information about all input and return parameters for the stored procedure.

Comments

- Table 2-13 describes the results set:

Column	Datatype	Description
procedure_qualifier	*varchar(30)*	
procedure_owner	*varchar(30)*	
procedure_name	*varchar(41)*	NOT NULL.
column_name	*varchar(30)*	NOT NULL.
column_type	*smallint*	

Table 2-13: Results Set for sp_sproc_columns

Column	Datatype	Description
data_type	*smallint*	The integer code for an ODBC datatype. If this datatype cannot be mapped into an ANSI type, the value will be NULL. The native datatype name is returned in the *type_name* column.
type_name	*char(30)*	The string representation of the datatype. This is the datatype name as presented by the underlying DBMS.
precision	*int*	The number of significant digits.
length	*int*	The length in bytes of the datatype.
scale	*smallint*	The number of digits to the right of the decimal point.
radix	*smallint*	Base for numeric types.
nullable	*smallint*	The value 1 means this datatype can be created allowing null values; 0 means it cannot.
remarks	*varchar(254)*	NULL
ss_data_type	*tinyint*	A SQL Server datatype.
colid	*tinyint*	A SQL Server specific column appended to the result set.

Table 2-13: Results Set for sp_sproc_columns (continued)

Messages

- `Table qualifier must be name of current database.`

  sp_sproc_columns cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_sproc_columns**.

Permissions

Any user can execute **sp_sproc_columns**.

Tables Used

sybsystemprocs..spt_datatype_info, syscolumns, sysobjects, sysprocedures, systypes

sp_statistics

Function

Returns a list of indexes on a single table.

Syntax

```
sp_statistics table_name [, table_owner]
   [, table_qualifier] [, index_name] [, is_unique]
```

Parameters

table_name – is the name of the table. No wildcard pattern matching is supported.

table_owner – is the owner of the table. No wildcard pattern matching is supported. If the parameter is not specified, **sp_statistics** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

index_name – is the index name. No wildcard pattern matching is supported.

is_unique – is Y if unique indexes are to be returned; otherwise, it is N.

Comments

- The indexes in the results set appear in ascending order by the columns *non-unique, type, index_name,* and *seq_in_index*.
- The index type *hashed* accepts exact match or range searches, but searches involving pattern matching do not use the index.
- Table 2-14 describes the results set:

Column	Datatype	Description
table_qualifier	*varchar*(32)	The database name. This field can be NULL.
table_owner	*varchar*(32)	
table_name	*varchar*(32)	NOT NULL.

Table 2-14: Results Set for sp_statistics

Column	Datatype	Description
non_unique	*smallint*	NOT NULL. The value 0 means unique, and 1 means not unique.
index_qualifier	*varchar*(32)	
index_name	*varchar*(32)	
type	*smallint*	NOT NULL. The value 0 means statistics for a table means clustered, 2 means hashed, and 3 means other.
seq_in_index	*smallint*	NOT NULL.
column_name	*varchar*(32)	NOT NULL.
collation	*char*(1)	The value A means ascending; D means descending; and NULL means not applicable.
cardinality	*int*	Number of rows in the table or unique values in the index.
pages	*int*	Number of pages to store the index or table.

Table 2-14: Results Set for sp_statistics (continued)

Messages

- `Table qualifier must be name of current database.`

  sp_statistics cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_statistics**.

- `Catalog procedure sp_statistics can not be run in a transaction.`

  sp_statistics modifies system tables, so it cannot be run within a transaction.

Permissions

Any user can execute **sp_statistics**.

Tables Used

syscolumns, sysindexes, sysobjects

sp_stored_procedures

Function

Returns information about one or more stored procedures.

Syntax

```
sp_stored_procedures [sp_name] [, sp_owner]
   [, sp_qualifier]
```

Parameters

sp_name – is the name of the stored procedure. Use wildcards to request information about more than one stored procedure.

sp_owner – is the owner of the stored procedure. Use wildcards to request information about procedures owned by more than one user.

sp_qualifier – is the name of the database. This can be the current database, or NULL.

Comments

- **sp_stored_procedures** can return the name of stored procedures for which the current user does not have execute permission.
- If the server attribute *accessible_sproc* is Y in the results set for **sp_server_info,** only stored procedures that are executable by the current user are returned.
- **sp_stored_procedures** only returns information about local stored procedures.
- Table 2-15 shows the results set:

Column	Datatype	Description
procedure_qualifier	*varchar(30)*	The name of the database.
procedure_owner	*varchar(30)*	
procedure_name	*varchar(41)*	NOT NULL.
num_input_params	*int*	NOT NULL.The value -1 means indeterminate, >= 0 means the number of parameters.

Table 2-15: Results Set for sp_stored_procedures

Column	Datatype	Description
num_output_params	*int*	NOT NULL. The value -1 means indeterminate, >= 0 means the number of parameters.
num_result_sets	*int*	NOT NULL. The value -1 means indeterminate, 0 means uses input/output parameters only, and > 0 means the number of results sets.
remarks	*varchar(254)*	NULL

Table 2-15: Results Set for sp_stored_procedures (continued)

Messages

- `Stored procedure qualifier must be name of current database.`

  sp_stored_procedures cannot be used to return information about stored procedures in another database. Execute the **use** command to switch to the desired database, then rerun **sp_stored_procedures**.

Permissions

Any user can execute **sp_stored_procedures**.

Tables Used

sysobjects, sysprocedures, sysprotects, sysusers

sp_table_privileges

Function

Returns privilege information for all columns in a table or view.

Syntax

```
sp_table_privileges table_name [, table_owner
   [, table_qualifier]]
```

Parameters

table_name – is the name of the table. No wildcard pattern matching is supported.

table_owner – is the name of the table owner. No wildcard pattern matching is supported. If you do not specify the table owner, **sp_table_privileges** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

Comments

- Table 2-16 shows the results set:

Column	Datatype	Description
table_qualifier	*varchar(32)*	The name of the database. This field can be NULL.
table_owner	*varchar(32)*	
table_name	*varchar(32)*	NOT NULL.
grantor	*varchar(32)*	NOT NULL.
grantee	*varchar(32)*	NOT NULL.

Table 2-16: Results Set for sp_table_privileges

Column	Datatype	Description
privilege	*varchar(32)*	Identifies the table privilege. May be one of the following: SELECT - The grantee is permitted to retrieve data for one or more columns of the table. INSERT - The grantee is permitted to insert new rows containing data for one or more columns into the table. UPDATE - The grantee is permitted to update the data in one or more columns of the table. DELETE - The grantee is permitted to delete rows of data from the table. REFERENCE - The grantee is permitted to refer to one or more columns of the table within a constraint.
is_grantable	*varchar(3)*	Indicates whether the grantee is permitted to grant the privilege to other users. The values are YES, NO, or NULL.

Table 2-16: Results Set for sp_table_privileges (continued)

Messages

- `Catalog procedure sp_table_privileges can not be run in a transaction.`

  sp_table_privileges updates system tables, so it cannot be run from within a transaction.

- `Object name can only be qualified with owner name.`
- `Object name must be qualified with the owner name.`
- `This may be a temporary object. Please execute procedure from tempdb.`

  You invoked **sp_table_priviliges** for a table beginning with "#". Execute the **use** command to switch to *tempdb*, then rerun **sp_table_priviliges**.

- `The table or view named doesn’t exist in the current database.`

  The table does not exist in the current database as specified. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_table_privileges cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_table_privileges**.

Permissions

Any user can execute **sp_table_privileges**.

Tables Used

sysobjects, sysusers

sp_tables

Function

Returns a list of objects that can appear in a **from** clause.

Syntax

```
sp_tables [table_name] [, table_owner]
   [, table_qualifier][, table_type]
```

Parameters

table_name – is the name of the table. Use wildcards to request information about more than one table.

table_owner – is the table owner. Use wildcards to request information about more than one table.

table_qualifier – is the name of the database. Acceptable values are the name of the current database and NULL.

table_type – A list of values, separated by commas, giving information about all tables of the table type(s) specified, including the following:

"'TABLE', 'SYSTEM TABLE', 'VIEW'"

➤ ***Note***

Enclose each table type with single quotation marks, and enclose the entire parameter with double quotation marks. Enter table types in uppercase.

Examples

1. **sp_tables @table_type = "'TABLE', 'VIEW'"**

   This procedure returns information about all tables in the current database of the type TABLE and VIEW and excludes information about system tables.

Comments

- SQL Server doesn't necessarily check your read and write permissions on *table_name*. Access to the table is not guaranteed, even if you can display information about it.

- The results set includes tables, views, and synonyms and aliases for gateways to DBMS products that support those types.
- If the server attribute *accessible_tables* is Y in the results set for **sp_server_info**, only tables that are accessible by the current user are returned.
- Table 2-17 shows the results set:

Column	Datatype	Description
table_qualifier	*varchar*(30)	The database name. This field can be NULL.
table_owner	*varchar*(30)	
table_name	*varchar*(30)	NOT NULL. The table name.
table_type	*varchar*(32)	NOT NULL. One of the following: 'TABLE', 'VIEW', 'SYSTEM TABLE'.
remarks	*varchar*(254)	NULL

Table 2-17: Results Set for sp_tables

Messages

- `Table qualifier must be name of current database.`

  sp_tables cannot be used to return information about tables in another database. Execute the **use** command to switch to the desired database, then rerun **sp_tables**.

Permissions

Any user can execute **sp_tables**.

Tables Used

sysdatabases, sysobjects, sysprotects, sysusers

Appendixes

A Reserved Words

Keywords are words that have a special meaning. This appendix lists Transact-SQL, APT-SQL, and SQL92 keywords.

Transact-SQL Reserved Words

The following words are reserved by SQL Server as keywords (command verbs) and cannot be used for the names of database objects such as databases, tables, rules, and defaults. Reserved words can be used for the names of local variables and for stored procedure parameter names.

add	close	disk	from
all	clustered	distinct	goto
alter	commit	double	grant
and	compute	drop	group
any	confirm	dummy	having
arith_overflow	constraint	dump	holdlock
as	continue	else	identity
asc	controlrow	end	identity_insert
at	convert	endtran	if
authorization	count	errlvl	in
avg	create	errorexit	index
begin	current	escape	insert
between	cursor	except	intersect
break	data_pgs	exec	into
browse	database	execute	is
bulk	dbcc	exists	isolation
by	deallocate	exit	key
cascade	declare	fetch	kill
char_convert	default	fillfactor	level
check	delete	for	like

Table A-1: Transact-SQL reserved words

checkpoint	desc	foreign	lineno
load	perm	rollback	to
max	permanent	rowcnt	tran
min	plan	rowcount	transaction
mirror	precision	rows	trigger
mirrorexit	prepare	rule	truncate
national	primary	save	tsequal
noholdlock	print	schema	union
nonclustered	privileges	select	unique
not	proc	set	update
null	procedure	setuser	used_pgs
numeric_truncation	processexit	shared	user
of	public	shutdown	user_option
off	raiserror	some	using
offsets	read	statistics	values
on	readtext	stripe	varying
once	reconfigure	sum	view
only	references	syb_identity	waitfor
open	replace	syb_restree	where
option	reserved_pgs	table	while
or	return	temp	with
order	revoke	temporary	work
over	role	textsize	writetext

Table A-1: Transact-SQL reserved words (continued)

APT-SQL Keywords

Table A-2: APT-SQL keywords lists the APT-SQL keywords which are not reserved words in Transact-SQL. If you are planning to use APT-SQL, avoid using these words as identifiers.

$channel	charindex	int	smallint
$curfield	closesql	interruptsql	sqlbegin
$curform	connect	list	sqlend
$curgroup	curindex	local	sqlexpr
$curpick	cursqlindex	log	sqlrow
$date	datalength	lower	submit
$index	datename	mchoice	substring
$status	datepart	menu	switch
abort	datetime	menubar	switchend
and	define	money	system
append	disconnect	nextquery	tab
apt	enter	nomsg	text
backtab	entry	opensql	textport
bell	exitform	parentname	tinyint
binary	exp	perform	trace
bit	false	positionform	transfer
call	fetchsql	post	trim
callextern	field	printform	true
callform	float	rchoice	upper
callreport	foreach	remote	useform
cancelform	form	reset	variable
case	global	schoice	
channel	hidden	scroll	
char	image	shared	

Table A-2: APT-SQL keywords

SQL92 Keywords

SQL Server 10.0 includes entry-level SQL92 features. Full SQL92 implementation includes the words listed in the following tables as command syntax. Since upgrading identifiers can be a complex process, we are providing this list for your convenience. The publication of this information does not commit Sybase to providing all of these SQL92 features in subsequent releases, and in addition subsequent releases may include keywords not included in this list.

Table A-3 lists the SQL92 keywords which are not reserved words in Transact-SQL.

absolute	constraints	false
action	corresponding	first
allocate	cross	float
are	current_date	found
assertion	current_time	full
bit	current_timestamp	get
bit_length	current_user	global
both	date	go
cascaded	day	hour
case	dec	immediate
cast	decimal	indicator
catalog	deferrable	initially
char	deferred	inner
character	describe	input
char_length	descriptor	insensitive
character_length	diagnostics	int
coalesce	disconnect	integer
collate	domain	interval
collation	end-exec	join
column	exception	language
connect	external	last
connection	extract	leading

Table A-3: SQL92 keywords that are not Transact-SQL reserved words

left	preserve	time
local	prior	timestamp
lower	real	timezone_hour
match	relative	timezone_minute
minute	restrict	trailing
module	right	translate
month	scroll	translation
names	second	trim
natural	section	true
nchar	session	unknown
next	session_user	upper
no	size	usage
nullif	smallint	value
numeric	space	varchar
octet_length	sql	when
outer	sqlcode	whenever
output	sqlerror	write
overlaps	sqlstate	year
pad	substring	zone
partial	system_user	
position	then	

Table A-3: SQL92 keywords that are not Transact-SQL reserved words (continued)

Potential SQL92 Reserved Words

If you are using the ISO / IEC 9075:1989 standard, also avoid using the words in *Table A-4: Potential reserved words*, as these words may become SQL92 reserved words in the future.

after	modify	routine
alias	new	row
async	none	savepoint
before	object	search
boolean	oid	sensitive
breadth	old	sequence
completion	operation	signal
call	operators	similar
cycle	others	sqlexception
data	parameters	structure
depth	pendant	test
dictionary	preorder	there
each	private	type
elseif	protected	under
equals	recursive	variable
general	ref	virtual
ignore	referencing	visible
leave	resignal	wait
less	return	without
limit	returns	
loop	role	

Table A-4: Potential reserved words

B The System Tables

Introduction

All of the tables in the *master* database are system tables. Some of these tables also occur in user databases—they are automatically created when the **create database** command is issued.

These system tables occur in all databases:

System Table	Contents
sysalternates	One row for each SQL Server user mapped to a database user
syscolumns	One row for each column in a table or view, and for each parameter in a procedure
syscomments	One or more rows for each view, rule, default, trigger, and procedure, giving SQL definition statement
sysconstraints	One row for each referential and check constraint associated with a table or column
sysdepends	One row for each procedure, view, or table that is referenced by a procedure, view, or trigger
sysindexes	One row for each clustered or nonclustered index, and one row for each table with no indexes, and an additional row for each table containing *text* or *image* data.
syskeys	One row for each primary, foreign, or common key; set by user (not maintained by SQL Server)
syslogs	Transaction log
sysobjects	One row for each table, view, procedure, rule, trigger default, log, and (in *tempdb* only) temporary object
sysprocedures	One row for each view, rule, default, trigger, and procedure, giving internal definition
sysprotects	User permissions information
sysreferences	One row for each referential integrity constraint declared on a table or column
sysroles	Maps server-wide roles to local database groups
syssegments	One row for each segment (named collection of disk pieces)

Table B-1: System tables that occur in all databases

System Table	Contents
systhresholds	One row for each threshold defined for the database
systypes	One row for each system-supplied and user-defined datatype
sysusermessages	One row for each user-defined message
sysusers	One row for each user allowed in the database

Table B-1: System tables that occur in all databases (continued)

These system tables occur in the *master* database only:

System Table	Contents
syscharsets	One row for each character set or sort order
sysconfigures	One row for each user-settable configuration parameter
syscurconfigs	Information about configuration parameters currently being used by SQL Server
sysdatabases	One row for each database on SQL Server
sysdevices	One row for each tape dump device, disk dump device, disk for databases, and disk partition for databases
sysengines	One row for each SQL Server engine currently on line
syslanguages	One row for each language (except U.S. English) known to the server
syslocks	Information about active locks
sysloginroles	One row for each server login that possesses a system-defined role
syslogins	One row for each valid SQL Server user account
sysmessages	One row for each system error or warning
sysprocesses	Information about server processes
sysremotelogins	One row for each remote user
syssrvroles	One row for each server-wide role
sysservers	One row for each remote SQL Server
sysusages	One row for each disk piece allocated to a database

Table B-2: System tables that occur in the master database only

These system tables occur in the *sybsecurity* database only:

System Table	Contents
sysaudits	One row for each audit record
sysauditoptions	One row for each global audit option

Table B-3: System tables that occur in the sybsecurity *database only*

In the pages that follow, each system table is described in more detail, including a list of their columns and datatypes. In addition, the indexes and the system procedures that reference a particular table are listed.

The word "reserved" in the column description means that the column is currently not being used by SQL Server.

Permissions for use of the system tables can be controlled by the database owner, just like permissions on any other tables.

The SYBASE installation program sets up permissions so that all users can read the system tables, with the exception of a few fields. (See the *SQL Server Installation Guide* for details.)

All direct updates on system tables are by default not allowed even for the database owner. Instead, SQL Server supplies system procedures to make any normally needed updates and additions to system tables.

You can allow direct updates to the system tables if it becomes necessary to modify them in a way that cannot be accomplished with a system procedure. To accomplish this, a System Security Officer must reset the configuration variable called **allow updates** with the system procedure **sp_configure**, and a System Administrator must then execute the **reconfigure** command. For information, see the *System Administration Guide*.

There are entries in some of the *master* database tables that should not be altered by any user under any circumstances. For example, do not attempt to modify *syslogs* with a **delete**, **update**, or **insert** command. In addition, an attempt to **delete** all rows from *syslogs* will put SQL Server into an infinite loop that eventually fills up the entire database.

Note that aggregate functions cannot be used on virtual tables such as *syslocks* and *sysprocesses*.

A diagram of the system tables and their relationships is included at the back of the bound *System Administration Guide* and the *SQL ServerReference Manuals* Volumes 1 and 2. The diagram is not included with camera-ready copy or CD-ROM versions of the manuals.

sysalternates

(all databases)

Description

sysalternates contains one row for each SQL Server user mapped (or aliased) to a user of the current database. When a user tries to access a database, SQL Server looks for a valid *uid* entry in *sysusers*. If none is found, it looks in *sysalternates.suid*. If the user's *suid* is found there, he or she is treated as the database user whose *suid* is listed in *sysalternates.altsuid*.

On the SQL Server distribution tape, there are no entries in *sysalternates*.

Column	Datatype	Description
suid	*smallint*	Server user ID of user being mapped
altsuid	*smallint*	Server user ID of user to whom another user is mapped

Table B-4: Columns in the sysalternates *table*

Indexes

Unique clustered index on *suid*

Referenced by System Procedures

sp_addalias, sp_adduser, sp_changedbowner, sp_dropalias, sp_dropuser, sp_helpuser

sysauditoptions

(*sybsecurity* database)

Description

sysauditoptions contains one row for each global audit option (options set via **sp_auditoption**).These are the system-wide options only, and do not include database, object, stored procedure, trigger, and user audit options. The default value for each option is 0 or "off." *sysauditoptions* can be accessed only by System Security Officers.

Column	Datatype	Description
optn	*smallint*	Option number. See Table B-6.
value	*smallint*	Current value; one of the following: **off** = 0 **ok** = 1 **fail** = 2 **both** = 3 (where applicable) For error auditing (*optn*=13), the values are: **off** = 0 **nonfatal** = 1 **fatal** = 2 **both** = 3
min	*smallint*	Minimum valid value for this option
max	*smallint*	Maximum valid value for this option
name	*varchar(30)*	Name of option
svalue	*varchar(30)*	String equivalent of the current value: for example, "on", "off", "nonfatal"
comment	*varchar(255)*	Description of option

Table B-5: Columns in the sysauditoptions *table*

Possible values for *optn* are:

Option Number	Description
1	Enable or disable auditing
2	Unused
3	Login auditing

Table B-6: Audit option values and descriptions

Option Number	Description
4	Logout auditing
5	Server boot auditing
6	RPC connection auditing
7	Auditing use of the **set** command to turn roles on and off
8	Auditing commands requiring **sa_role** role
9	Auditing commands requiring **sso_role** role
10	Auditing commands requiring **oper_role** role
12	Auditing commands requiring **navigator** role
13	Error auditing
14	Ad hoc auditing
15	Auditing commands requiring **replication** role

Table B-6: Audit option values and descriptions (continued)

Indexes

None

Referenced by System Procedures

sp_auditoption

sysaudits

(*sybsecurity* database)

Description

The *sysaudits* table contains one row for each audit record.

Column	Datatype	Description
event	*smallint*	Type of event being audited. See Table B-8.
eventmod	*smallint*	Further information about the event. Possible values are: 0 = no modifier for this event 1 = successful occurrence of this event; for error auditing (*event*=13), a nonfatal error 2 = failed occurrence of this event; for error auditing (*event*=13), a fatal error
spid	*smallint*	Server process ID of the process that caused the audit record to be written
eventtime	*datetime*	Date and time of the audited event
sequence	*smallint*	Sequence number of the record within a single event; some events require more than one audit record
suid	*smallint*	Server login ID of the user who performed the audited event
dbid	*int null*	Database ID in which the audited event occurred or the object / stored procedure / trigger resides, depending on the type of event
objid	*int null*	ID of the accessed object or stored procedure / trigger
xactid	*binary(6) null*	ID of the transaction containing the audited event. For a multi-database transaction, this is the transaction ID from the database where the transaction originated.
loginname	*varchar(30) null*	Login name corresponding to the *suid*
dbname	*varchar(30) null*	Database name corresponding to the *dbid*
objname	*varchar(30) null*	Object name corresponding to the *objid*

Table B-7: Columns in the sysaudits *table*

Column	Datatype	Description
objowner	*varchar(30) null*	Name of the owner of *objid*
extrainfo	*varchar(255) null*	Additional information about the audited event; contents vary with the type of event audited. (See Table B-8.)

Table B-7: Columns in the sysaudits *table*

Possible values for the *event* column are shown in Table B-8, along with the corresponding contents of the *extrainfo* column. Global audit events have a code of less than 100. All other event types are numbered starting at 100.

event No.	Description	Contents of *extrainfo* Column
1	Enable auditing	NULL
2	Disable auditing	NULL
3	Login	Host name
4	Logout	Host name
5	Server boot	Names of the server program, master device, interfaces file path, server, and error log file
6	RPC connection	Remote server name, host name
7	Use of **set** command to turn roles on and off	Role, new setting
8	Command requiring **sa_role** role	Command type
9	Command requiring **sso_role** role	Command type
10	Command requiring **oper_role** role	Command type
12	Command requiring **navigator** role	Command type
13	Error	Error number, severity, and state
14	Ad hoc audit record	User-supplied comment text
15	Command requiring **replication** role	Command type
100	Database reference	Command type
101	Table reference	Command type
102	View reference	Command type
103	Stored procedure execution	Parameter list
104	Trigger execution	NULL
105	User's attempts to access a table	Command type
106	User's attempt to access a view	Command type
107	User's command text auditing	Command batch text

Table B-8: Contents of event *and* extrainfo *columns of* sysaudits

Indexes

None

Referenced by System Procedures

None

syscharsets

(*master* database only)

Description

syscharsets contains one row for each character set and sort order defined for use by SQL Server. One of the sort orders is marked in *master..sysconfigures* as the default sort order, which is the only one actually in use.

Column	Datatype	Description
type	*smallint*	The type of entity this row represents. Numbers from 1001 to 1999 represent character sets. Numbers from 2000 to 2999 represent sort orders.
id	*tinyint*	The ID for a character set or sort order. A sort order is defined by the combination of the sort order ID and the character set ID (*csid*). The character set is defined by id, which must be unique. Sybase reserves ID numbers 0-200.
csid	*tinyint*	If the row represents a character set, this field is unused. If the row represents a sort order, this is the ID of the character set that sort order is built on. A character set row with this ID must exist in this table.
status	*smallint*	Internal system status information bits.
name	*varchar(30)*	A unique name for the character set or sort order. Must contain only the 7-bit ASCII letters A-Z or a-z, digits 0-9, and underscores (_), and begin with a letter.
description	*varchar(255)*	An optional description of the features of the character set or sort order.
definition	*image*	The internal definition of the character set or sort order. The structure of the data in this field depends on the *type*.

Table B-9: Columns in the syscharsets *table*

Indexes

unique clustered index on *id*, *csid*, *type*
unique nonclustered index on *name*

Referenced by System Procedures

sp_checkreswords, sp_helpsort, sp_serverinfo

syscolumns

(all databases)

Description

syscolumns contains one row for every column in every table and view, and a row for each parameter in a procedure.

Column	Datatype	Description
id	*int*	ID of table to which this column belongs or of procedure with which this parameter is associated
number	*smallint*	Sub-procedure number when the procedure is grouped (0 for non-procedure entries)
colid	*tinyint*	Column ID
status	*tinyint*	Indicates unique position for *bit* columns, whether NULL values are legal in this column, and if a check constraint exists for the column
type	*tinyint*	Physical storage type; copied from *systypes*
length	*tinyint*	Physical length of data; copied from *systypes* or supplied by user
offset	*smallint*	Offset into the row where this column appears, if negative, this is a variable-length column
usertype	*smallint*	User type ID; copied from *systypes*
cdefault	*int*	ID of the procedure that generates default value for this column
domain	*int*	Constraint ID of the first rule or check constraint for this column
name	*sysname*	Column name
printfmt	*varchar(255)*	Reserved
prec	*tinyint*	Number of significant digits
scale	*tinyint*	Number of digits to the right of the decimal point

Table B-10: Columns in the syscolumns *table*

Indexes

unique clustered index on *id, number, colid*

Referenced by System Procedures

sp_bindefault, sp_bindrule, sp_changegroup, sp_checknames, sp_checkreswords, sp_column_privileges, sp_columns, sp_commonkey, sp_droptype, sp_dropuser, sp_estspace, sp_foreignkey, sp_help, sp_helpconstraint, sp_helpjoins, sp_helprotect, sp_primarykey, sp_rename, sp_special_columns, sp_sproc_columns, sp_statistics, sp_unbindefault, sp_unbindrule

syscomments

(all databases)

Description

syscomments contains entries for each view, rule, default, trigger, table constraint, and procedure. The *text* field contains the original definition statements. If the *text* is longer than 255 bytes, the entries will span rows. Each object can occupy up to 65025 rows.

Column	Datatype	Description
id	*int*	Object ID to which this text applies
number	*smallint*	Sub-procedure number when the procedure is grouped (0 for non-procedure entries)
colid	*tinyint*	Sequence of 255 rows for the object
texttype	*smallint*	0 for system-supplied comment (for views, rules, defaults, triggers, and procedures). 1 for user-supplied comment (users can add entries that describe an object or column)
language	*smallint*	Reserved
text	*varchar(255)*	Actual text of SQL definition statement
colid2	*tinyint*	Indicates next sequence of rows for the object (see *colid* above); object can have up to 255 sequences of 255 rows each.

Table B-11: Columns in the syscomments *table*

Indexes

unique clustered index on *id, number, colid, texttype*

Referenced by System Procedures

sp_helpconstraint, sp_helptext

sysconfigures

(*master* database only)

Description

sysconfigures and *syscurconfigs* contain one row for each user-settable configuration variable.

Column	Datatype	Description
config	*smallint*	Configuration variable number
value	*int*	The user-modifiable value for the variable (being used by SQL Server only if **reconfigure** has been run)
comment	*varchar(255)*	Explanation of the configuration variable
status	*smallint*	Either 1 (dynamic, meaning variable takes effect when **reconfigure** is issued) or 0 (variable takes effect when SQL Server is restarted

Table B-12: Columns in the sysconfigures *table*

Here are the contents of *sysconfigures:*

config	value	comment	status
101	0	Maximum recovery interval in minutes	1
102	1	Allow updates to system tables	1
103	0	Number of user connections allowed	0
104	0	Size of available physical memory in 2k pages	0
105	0	Number of open databases allowed among all users	0
106	0	Number of locks for all users	0
107	0	Number of open database objects	0
108	0	Percentage of remaining memory used for procedure cache	0
109	0	Default fill factor percentage	0
110	0	Average time slice per process in milliseconds	0
111	0	Default database size in megabytes	0
112	0	Tape retention period in days	0
113	0	Recovery flags	0
115	1	Allow triggers to be invoked within triggers	1
116	0	Number of devices	0

Table B-13: Contents of sysconfigures

config	value	comment	status
117	1	Allow remote access	1
118	0	Number of remote logins	0
119	0	Number of remote sites	0
120	0	Number of remote connections	0
121	0	Number of pre-read packets per remote connection	0
122	1001	Upgrade version	1
123	50	Default sortorder ID	0
124	0	Default language	1
125	3	Language cache	0
126	1	Maximum online engines	0
127	1	Minimum online engines	0
128	0	Engine adjust interval	0
129	200	CPU accounting flush interval	1
130	1000	I/O accounting flush interval	1
131	1	Default character set ID	0
134	0	Stack size	0
135	0	System-wide password expiration interval	1
136	100	Audit queue size	0
137	0	Additional netmem	0
138	0	Default network packet size	0
139	0	Maximum network packet size	0
140	0	Number of extent I/O buffers	0
141	5000	Identity burning set factor	0

Table B-13: Contents of sysconfigures *(continued)*

Indexes

unique clustered index on *config*

Referenced by System Procedures

sp_configure

sysconstraints

(all databases)

Description

The *sysconstraints* table has one row for each referential and check constraint associated with a table or column.

Whenever a user declares a new check constraint or referential constraint using **create table** or **alter table**, SQL Server inserts a row into the *sysconstraints* table. The row remains until a user executes **alter table** to drop the constraint. Dropping a table by executing **drop table** removes all rows associated with that table from the *sysconstraints* table.

Column	Datatype	Description
colid	*tinyint*	Column number in the table
spare1	*tinyint*	Unused
constrid	*int*	Object ID of the constraint
tableid	*int*	ID of the table on which the constraint is declared
error	*int*	Constraint specific error message
status	*int*	The type of constraint: 0x0040 = a referential constraint 0x0080 = a check constraint
spare2	*int*	Unused

Table B-14: Columns in the sysconstraints *table*

Indexes

clustered index on *tableid, colid*
unique nonclustered index on *constrid*

Referenced by System Procedures

sp_bindmsg, sp_bindrule, sp_helpconstraint, sp_unbindmsg, sp_unbindrule

syscurconfigs

(*master* database only)

Description

syscurconfigs is built dynamically when queried. Its structure is identical to that of *sysconfigures*. It contains an entry for each of the configuration variables, as does *sysconfigures*, but with the current values rather than the default values. In addition, it contains four rows that describe the configuration structure.

Column	Datatype	Description
config	*smallint*	Configuration variable number
value	*int*	The user-modifiable value for the variable (being used by SQL Server only if **reconfigure** has been run)
comment	*varchar(255)*	Explanation of the configuration variable
status	*smallint*	Either 1 (dynamic, meaning variable takes effect when **reconfigure** is issued) or 0 (variable takes effect when SQL Server is restarted

Table B-15: Columns in the syscurconfigs *table*

Referenced by System Procedures

sp_configure, sp_helpsort, sp_serverinfo

sysdatabases

(*master* database only)

Description

sysdatabases contains one row for each database on SQL Server. When SQL Server is installed, *sysdatabases* contains entries for the *master* database, the *model* database, the *sybsystemprocs* database and the *temporary* database. If you have installed auditing, it also contains an entry for the *sybsecurity* database

Column	Datatype	Description
name	*sysname*	Name of the database
dbid	*smallint*	Database ID
suid	*smallint*	Server user ID of database creator
status	*smallint*	Control bits; those which the user can set with **sp_dboption** are marked “settable.” See Table B-17.
version	*smallint*	Version of SQL Server code under which database was created
logptr	*int*	Pointer to transaction log
crdate	*datetime*	Creation date
dumptrdate	*datetime*	Date of the last **dump transaction**
status2	*intn*	Additional control bits. See Table B-18.
audflags	*intn*	Audit settings for database
deftabaud	*intn*	Bit-mask that defines default audit settings for tables
defvwaud	*intn*	Bit-mask that defines default audit settings for views
defpraud	*intn*	Bit-mask that defines default audit settings for stored procedures

Table B-16: Columns in the sysdatabases *table*

The bit representations for the *status* column are:

Decimal	Hex	Status
4	0x04	**select into/bulkcopy**; settable
8	0x08	**trunc log on chkpt**; settable
16	0x10	**no chkpt on recovery**; settable
32	0x20	crashed while loading database, instructs recovery not to proceed
256	0x100	database suspect; not recovered; cannot be opened or used; can only be dropped with **dbcc dbrepair**
512	0x200	**ddl in tran**; settable
1024	0x400	**read only**; settable
2048	0x800	**dbo use only**; settable
4096	0x1000	**single user**; settable
8192	0x2000	**allow nulls by default**; settable
16384	0x4000	*dbname* has changed

Table B-17: status *control bits in the* sysdatabases *table*

The bit representations for the *status2* column are:

Decimal	Hex	Status
1	0x01	**abort tran on log full**; settable
2	0x02	**no free space acctg**; settable
4	0x04	**auto identity**; settable

Table B-18: status2 *control bits in the* sysdatabases *table*

Indexes

unique clustered index on *name*
unique nonclustered index on *dbid*

Referenced by System Procedures

sp_addlogin, sp_addsegment, sp_addtype, sp_changedbowner, sp_checknames, sp_checkreswords, sp_databases, sp_dboption, sp_dbremap, sp_dropdevice, sp_dropsegment, sp_extendsegment, sp_helpdb, sp_logdevice, sp_renamedb, sp_tables

sysdepends

(all databases)

Description

sysdepends contains one row for each procedure, view, or table that is referenced by a procedure, view, or trigger.

Column	Datatype	Description
id	*int*	Object ID
number	*smallint*	Procedure number
depid	*int*	Dependent object ID
depnumber	*smallint*	Dependent procedure number
status	*smallint*	Internal status information
selall	*bit*	On if object is used in **select** * statement
resultobj	*bit*	On if object is being updated
readobj	*bit*	On if object is being read

Table B-19: Columns in the sysdepends *table*

Indexes

unique clustered index on *id, number, depid, depnumber*

Referenced by System Procedures

sp_depends

sysdevices

(*master* database only)

Description

sysdevices contains one row for each tape dump device, disk dump device, disk for databases, and disk partition for databases. On the SQL Server distribution tape, there are four entries in *sysdevices*: one for the master device (for databases), one for a disk dump device, and two for tape dump devices.

Column	Datatype	Description
low	*int*	First virtual page number on database device (not used for dump devices)
high	*int*	Last virtual page number on database device or dump device
status	*smallint*	Bit map indicating type of device, default and mirror status. See Table B-21.
cntrltype	*smallint*	Controller type (0 if database device, 2 if disk dump device or streaming tape, 3–8 if tape dump device)
name	*sysname*	Logical name of dump device or of database device
phyname	*varchar(127)*	Name of physical device
mirrorname	*varchar(127)*	Name of mirror device

Table B-20: Columns in the sysdevices *table*

The bit representations for the *status* column are additive. For example, "3" indicates a physical disk that is also a default.

The status control bits are:

Decimal	Hex	Status
1	0x01	default disk
2	0x02	physical disk
4	0x04	logical disk
8	0x08	skip header

Table B-21: status *control bits in the* sysdevices *table*

Decimal	Hex	Status
16	0x10	dump device
32	0x20	serial writes
64	0x40	device mirrored
128	0x80	reads mirrored
256	0x100	secondary mirror side only
512	0x200	mirror enabled
2048	0x800	used internally

Table B-21: status *control bits in the* sysdevices *table*

Indexes

unique clustered index on *name*

Referenced by System Procedures

sp_addsegment, sp_addumpdevice, sp_checknames, sp_checkreswords, sp_configure, sp_diskdefault, sp_dropdevice, sp_dropsegment, sp_extendsegment, sp_helpdb, sp_helpdevice, sp_helplog, sp_helpsegment, sp_logdevice, sp_volchanged

sysengines

(*master* database only)

Description

sysengines contains one row for each SQL Server engine currently on line.

Column	Datatype	Description
engine	*smallint*	Engine number
osprocid	*int*	Operating system process ID (may be NULL)
osprocname	*char*	Operating system process name (may be NULL)
status	*char*	One of: online, off-line, in create, in destroy, debug
affinitied	*int*	Number of SQL Server processes with affinity to this engine
cur_kpid	*int*	Kernel process ID of process currently running on this engine, if any
last_kpid	*int*	Kernel process ID of process which previously ran on this engine
idle_1	*tinyint*	Reserved
idle_2	*tinyint*	Reserved
idle_3	*tinyint*	Reserved
idle_4	*tinyint*	Reserved
starttime	*datetime*	Date and time engine came on line

Table B-22: Columns in the sysengines *table*

Indexes

none

Referenced by System Procedures

sp_monitor

sysindexes

(all databases)

Description

sysindexes contains one row for each clustered index, one row for each nonclustered index, one row for each table that has no clustered index, and one row for each table that contains *text* or *image* columns.

The column *doampg* is used only if the row describes a table or clustered index.

Column	Datatype	Description
name	*sysname*	Index or table name
id	*int*	ID of table, or ID of table to which index belongs
indid	*smallint*	0 if table, 1 if clustered index, >1 if nonclustered, 255 if text chain
doampg	*int*	Page number for the object allocation map of a table or clustered index
ioampg	*int*	Page number for the allocation map of a nonclustered index
oampgtrips	*int*	Ratio of OAM page to data page residency in cache
status2	*int*	Internal system status information. See Table B-24.
ipgtrips	*int*	Ratio of index page to data page residency in cache
first	*int*	Pointer to first data or leaf page
root	*int*	Pointer to root page if entry is an index; pointer to last page if entry is a table or text chain
distribution	*int*	Pointer to distribution page (if entry is an index)
usagecnt	*smallint*	Reserved
segment	*smallint*	Number of segment in which this object resides

Table B-23: Columns in the sysindexes *table*

Column	Datatype	Description
status	*smallint*	Internal system status information (See Table B-25)
rowpage	*smallint*	Maximum number of rows per page
minlen	*smallint*	Minimum size of a row
maxlen	*smallint*	Maximum size of a row
maxirow	*smallint*	Maximum size of a non-leaf index row
keycnt	*smallint*	Number of keys for a clustered index; number of keys+1 for a nonclustered index
keys1	*varbinary(255)*	Description of key columns (if entry is an index)
keys2	*varbinary(255)*	Description of key columns (if entry is an index)
soid	*tinyint*	Sort order ID that the index was created with. ‘0’ if there is no character data in the keys
csid	*tinyint*	Character set ID that the index was created with. ‘0’ if there is no character data in the keys

Table B-23: Columns in the sysindexes *table*

The bit representations for the *status2* column are:

Decimal	Hex	Status
1	0x1	Index part of referential integrity constraint
2	0x2	Index part of primary key / unique constraint

Table B-24: status2 *control bits in the* sysindexes *table*

The bit representations for the *status* column are:

Decimal	Hex	Status
1	0x1	Abort command or trigger if attempt to insert duplicate key
2	0x2	Unique index
4	0x4	Abort command or trigger if attempt to insert duplicate row
16	0x10	Clustered index
64	0x40	Index allows duplicate rows
32768	0x8000	Suspect index

Table B-25: status *control bits in the* sysindexes *table*

Indexes

unique clustered index on *id, indid*

Referenced by System Procedures

sp_checknames, sp_checkreswords, sp_dropsegment, sp_estspace, sp_help, sp_helpconstraint, sp_helpindex, sp_helplog, sp_helpsegment, sp_indsuspect, sp_pkeys, sp_placeobject, sp_rename, sp_spaceused, sp_special_columns, sp_statistics

syskeys

(all databases)

Description

syskeys contains one row for each primary, foreign, or common key.

Column	Datatype	Description
id	*int*	Object ID
type	*smallint*	Record type
depid	*int null*	Dependent object ID
keycnt	*int null*	The number of non-null keys
size	*int null*	Reserved
key1	*int null*	Column ID
key2	*int null*	Column ID
key3	*int null*	Column ID
key4	*int null*	Column ID
key5	*int null*	Column ID
key6	*int null*	Column ID
key7	*int null*	Column ID
key8	*int null*	Column ID
depkey1	*int null*	Column ID
depkey2	*int null*	Column ID
depkey3	*int null*	Column ID
depkey4	*int null*	Column ID
depkey5	*int null*	Column ID
depkey6	*int null*	Column ID
depkey7	*int null*	Column ID
depkey8	*int null*	Column ID

Table B-26: Columns in the syskeys *table*

Indexes

clustered index on *id*

Referenced by System Procedures

sp_commonkey, sp_dropkey, sp_foreignkey, sp_helpjoins, sp_helpkey, sp_primarykey

syslanguages

(*master* database only)

Description

syslanguages contains one row for each language known to SQL Server. *us_english* is not in *syslanguages*, but is always available to SQL Server.

Column	Datatype	Description
langid	*smallint*	Unique language ID
dateformat	*char(3)*	Date order, for example, "dmy"
datefirst	*tinyint*	First day of the week—1 for Monday, 2 for Tuesday, and so on, up to 7 for Sunday.
upgrade	*int*	SQL Server version of last upgrade for this language
name	*varchar(30)*	Official language name, for example, "french"
alias	*varchar(30)*	Alternate language name, for example, "francais"
months	*varchar(251)*	Comma-separated list of full-length month names, in order from January to December—each name is at most 20 characters long
shortmonths	*varchar(119)*	Comma-separated list of shortened month names, in order from January to December—each name is at most 9 characters long
days	*varchar(216)*	Comma-separated list of day names, in order from Monday to Sunday—each name is at most 30 characters long.

Table B-27: Columns in the syslanguages *table*

Indexes

unique clustered index on *langid*
unique nonclustered index on *name*
unique nonclustered index on *alias*

Referenced by System Procedures

sp_addlanguage, sp_addmessage, sp_checkreswords, sp_configure, sp_droplanguage, sp_dropmessage, sp_getmessage, sp_helplanguage, sp_setlangalias

syslocks

(*master* database only)

Description

syslocks contains information about active locks, but it is not a normal table. Rather, it is built dynamically when queried by a user. No updates to *syslocks* are allowed.

Column	Datatype	Description
id	*int*	Table ID
dbid	*smallint*	Database ID
page	*int*	Page number
type	*smallint*	Type of lock *(*bit values for the *type* column are listed in Table B-29)
spid	*smallint*	ID of process that holds the lock
class	*char(30)*	Name of the cursor this lock is associated with, if any

Table B-28: Columns in the syslocks *table*

The bit representations for the type column are:

Decimal	Hex	Status
1	0x1	Exclusive table lock
2	0x2	Shared table lock
3	0x3	Exclusive intent lock (will do page locking on indicated pages)
4	0x4	Shared intent lock
5	0x5	Exclusive page lock
6	0x6	Shared page lock
7	0x7	Update page lock (changes to exclusive if page is actually modified)
256	0x100	Lock is blocking another process
512	0x200	Demand lock

Table B-29: type *control bit in the* syslocks *table*

Indexes

none

Referenced by System Procedures

sp_lock

sysloginroles

(master database only)

Description

sysloginroles contains a row for each instance of a server login possessing a system-defined role. One row is added for each role possessed by each login. For example, if a single server user is granted three roles, three rows are added to *sysloginroles* associated with that user's *suid*.

Column	Datatype	Description
suid	*suid*	Server user ID
srid	*smallint*	Server role ID; one of the following: 0 **sa_role** 1 **sso_role** 2 **oper_role** 4 **navigator_role** 5 **replication_role** 6 **bcpin_labels_role**
status	*smallint*	Reserved

Table B-30: Columns in the sysloginroles *table*

Indexes

Clustered index on *suid*

Referenced by System Procedures

sp_displaylogin, sp_droplogin, sp _locklogin, sp_role

syslogins

(*master* database only)

Description

syslogins contains one row for each valid SQL Server user account. On the SQL Server distribution tape, *syslogins* contains an entry in which the name is "sa", the suid is 1, and the password is null. It also contains an entry named "probe" with an unpublished password. The login "probe" and the user "probe" exist for the Two Phase Commit Probe Process, which uses a challenge and response mechanism to access SQL Server.

Column	Datatype	Description
suid	*smallint*	Server user ID
status	*smallint*	Status of the account. See Table B-32.
accdate	*datetime*	Date *totcpu* and *totio* were last cleared
totcpu	*int*	CPU time accumulated by login
totio	*int*	I/O accumulated by login
spacelimit	*int*	Reserved
timelimit	*int*	Reserved
resultlimit	*int*	Reserved
dbname	*sysname*	Name of database in which to put user when connection established
name	*sysname*	Login name of user
password	*varbinary*	Password of user (encrypted)
language	*varchar(30)*	User's default language
pwdate	*datetime*	Date the password was last changed
audflags	*int*	User's audit settings
fullname	*varchar(30)*	Full name of the user

Table B-31: Columns in the syslogins *table*

The bit representations for the *status* column are:

Decimal	Hex	Status
1	0x1	Password less than 6 characters, or NULL
2	0x2	Account is locked
4	0x4	Password is expired

Table B-32: status *control bits in the* syslogins *table*

Indexes

unique clustered index on *suid*
unique nonclustered index on *name*

Referenced by System Procedures

sp_addalias, sp_addlogin, sp_addremotelogin, sp_adduser, sp_changedbowner, sp_checknames, sp_checkreswords, sp_clearstats, sp_displaylogin, sp_droplogin, sp_helpdb, sp_helpuser, sp_locklogin, sp_modifylogin, sp_reportstats, sp_role

syslogs

(all databases)

Description

syslogs contains the transaction log. It is used by SQL Server for recovery and roll forward, and is not useful to users.

You cannot delete from, insert into, or update *syslogs*. Every data modification operation is logged, so before you can change *syslogs*, the change must be logged. This means that a change operation on *syslogs* adds a row to *syslogs*, which then must be logged, adding another row to *syslogs*, and so on, producing an infinite loop. The loop continues until the database becomes full.

Column	Datatype	Description
xactid	*binary(6)*	Transaction ID
op	*tinyint*	Update operation number

Table B-33: Columns in the syslogs *table*

Indexes

none

sysmessages

(*master* database only)

Description

sysmessages contains one row for each system error or warning that can be returned by SQL Server. SQL Server displays the error description on the user's screen.

Column	Datatype	Description
error	*int*	Unique error number
severity	*smallint*	Severity level of error
dlevel	*smallint*	Reserved for number of descriptive level of this message: terse, short, or long
description	*varchar(25)*	Explanation of error with place holders for parameters
langid	*smallint*	Language, null for us_english
sqlstate	*varchar(5)*	SQLSTATE value for the error

Table B-34: Columns in the sysmessages *table*

Indexes

clustered index on *error, dlevel*
unique nonclustered index on *error, dlevel, langid*

Referenced by System Procedures

sp_configure, sp_dboption, sp_depends, sp_droplanguage, sp_getmessage, sp_help, sp_helpdb, sp_helpdevice, sp_helpremotelogin, sp_remoteoption

sysobjects

(all databases)

Description

sysobjects contains one row for each table, view, stored procedure, log, rule, default, trigger, check constraint, referential constraint, and (in *tempdb* only) temporary object.

Column	Datatype	Description
name	*sysname*	Object name
id	*int*	Object ID
uid	*smallint*	User ID of object owner
type	*char(2)*	One of the following object types: S system table U user table V view L log P procedure R rule D default TR trigger RI referential constraint
userstat	*smallint*	Application-dependent type information (32768 decimal [0x8000 hex] indicates to Data Workbench^{TM} that a procedure is a report)
sysstat	*smallint*	Internal status information (256 decimal [0x100 hex] indicates that table is read-only)
indexdel	*smallint*	Index delete count (incremented if an index is deleted)
schemacnt	*smallint*	Count of changes in schema of a given object (incremented if a rule or default is added)
sysstat2	*smallint*	Additional internal status information. See Table B-36.
crdate	*datetime*	Date object was created
expdate	*datetime*	Reserved
deltrig	*int*	Stored procedure ID of a delete trigger
instrig	*int*	Stored procedure ID of an insert trigger

Table B-35: Columns in the sysobjects *table*

Column	Datatype	Description
updtrig	*int*	Stored procedure ID of an update trigger
seltrig	*int*	Reserved
ckfirst	*int*	ID of first check constraint on the table
cache	*smallint*	Reserved
audflags	*int*	Object's audit settings
objspare	*int*	Spare

Table B-35: Columns in the sysobjects *table (continued)*

The bit representations for the *sysstat2* column are:

Decimal	Hex	Status
1	0x1	Table has referential constraint
2	0x2	Table has foreign key constraint
4	0x4	Table has more than one check constraint
8	0x8	Table has primary key constraint
16	0x10	Chained transaction mode only stored procedure
32	0x20	Any transaction mode stored procedure
64	0x40	Table has IDENTITY field

Table B-36: systat2 *control bits in the* sysobjects *table*

Indexes

unique clustered index on *id*
unique nonclustered index on *name, uid*

Referenced by System Procedures

sp_addmessage, sp_addthreshold, sp_bindefault, sp_bindmsg, sp_bindrule, sp_checknames, sp_checkreswords, sp_column_privileges, sp_columns, sp_commonkey, sp_depends, sp_dropgroup, sp_dropkey, sp_dropsegment, sp_dropthreshold, sp_droptype, sp_dropuser, sp_estspace, sp_fkeys, sp_foreignkey, sp_help, sp_helpconstraint, sp_helpindex, sp_helpjoins, sp_helpkey, sp_helprotect, sp_helpthreshold, sp_indsuspect, sp_modifythreshold, sp_pkeys, sp_placeobject, sp_primarykey, sp_procxmode, sp_recompile, sp_remap, sp_rename, sp_spaceused, sp_sproc_columns, sp_statistics, sp_stored_procedures, sp_table_privileges, sp_tables, sp_unbindefault, sp_unbindmsg, sp_unbindrule

sysprocedures

(all databases)

Description

sysprocedures contains entries for each view, default, rule, trigger, procedure, declarative default, and check constraint. The plan or sequence tree for each object is stored in binary form. If the sequence tree doesn't fit in one entry, it is broken into more than one row. The *sequence* column identifies the sub-rows.

Column	Datatype	Description
type	*smallint*	Object type. See Table B-38.
id	*int*	Object ID
sequence	*smallint*	Sequence number if more than one row is used to describe this object
status	*smallint*	Internal system status
number	*smallint*	Sub-procedure number when the procedure is grouped (0 for non-procedure entries)

Table B-37: Columns in the sysprocedures *table*

The bit representations for the *type* column are:

Decimal	Hex	Status
1	0x1	Entry describes a plan (reserved)
2	0x2	Entry describes a tree

Table B-38: type *control bits in the* sysprocedures *table*

Indexes

unique clustered index on *id, type, sequence, number*

Referenced by System Procedures

sp_bindefault, sp_bindrule, sp_remap, sp_sproc_columns, sp_stored_procedures, sp_unbindefault, sp_unbindrule

sysprocesses

(*master* database only)

Description

sysprocesses contains information about SQL Server processes, but it is not a normal table. Rather, it is built dynamically when queried by a user. No updates to *sysprocesses* are allowed.

Use the **kill** statement to kill a process.

Column	Datatype	Description
spid	*smallint*	Process ID
kpid	*int*	Kernel process ID
enginenum	*int*	Number of engine on which process is being executed
status	*char(12)*	Process ID status, one of: infected background recv sleep send sleep alarm sleep lock sleep sleeping runnable running stopped bad status log suspend
suid	*smallint*	Server user ID of user who issued command
hostname	*char(10)*	Name of host computer
program_name	*char(16)*	Name of front-end module
hostprocess	*char(8)*	Host process ID number
cmd	*char(16)*	Command currently being executed
cpu	*int*	Cumulative cpu time for process in ticks
physical_io	*int*	Number of disk reads and writes for current command
memusage	*int*	Amount of memory allocated to process
blocked	*smallint*	Process ID of blocking process, if any

Table B-39: Columns in the sysprocesses *table*

Column	Datatype	Description
dbid	*smallint*	Database ID
uid	*smallint*	ID of user who executed command
gid	*smallint*	Group ID of user who executed command
tran_name	*varchar*(64)	Name of the active transaction
time_blocked	*int*	Time blocked in seconds
network_pktsz	*int*	Current connection's network packet size

Table B-39: Columns in the sysprocesses *table*

Indexes

none

Referenced by System Procedures

sp_dboption, sp_droplogin, sp_locklogin, sp_role, sp_who

sysprotects

(all databases)

Description

sysprotects contains information on user permissions information—entries for each **grant** and **revoke** statement that has been issued.

Column	Datatype	Description
id	*int*	ID of object to which this permission applies
uid	*smallint*	ID of user or group to which this permission applies
action	*tinyint*	One of the following permissions: **select =** 193 **insert** = 195 **delete** = 196 **update** = 197 **execute** = 224 **references** = 151 **create database** = 203 **create default** = 233 **create procedure** = 222 **create rule** = 236 **create table** = 198 **create view** = 207 **dump database** = 228 **dump transaction** = 235
protecttype	*tinyint*	One of the following values: **grant with grant** = 0 **grant** (permanent) = 1 **revoke** (permanent) = 2
columns	*varbinary(32)*	Bit map of columns to which this **select** or update permission applies. Bit 0 indicates all columns; 1 means permission applies to that column; null means no information.
grantor	*smallint*	User ID of the grantor

Table B-40: Columns in the sysprotects *table*

Indexes

unique clustered index on *id, action, grantor, uid, protecttype*

Referenced by System Procedures

sp_changegroup, sp_dropgroup, sp_dropuser, sp_helprotect, sp_stored_procedures, sp_tables

sysreferences

(all databases)

Description

sysreferences contains one row for each referential integrity constraint declared on a table or column.

Column	Datatype	Description
indexid	*smallint*	ID of the unique index on referenced columns
constrid	*int*	Object ID of the constraint from *sysobjects*
tableid	*int*	Object ID of the referencing table
reftabid	*int*	Object ID of the referenced table
keycnt	*smallint*	The number of columns in the foreign key
status	*smallint*	Reserved
frgndbid	*smallint*	Reserved
pmrydbid	*smallint*	Reserved
spare2	*int*	Reserved
fokey1	*tinyint*	Column ID of the first referencing column
. . .		
fokey16	*tinyint*	Column ID of the 16th referencing column
refkey1	*tinyint*	Column ID of the first referenced column
. . .		
refkey16	*tinyint*	Column ID of the 16th referenced column
frgndbname	*varchar(30)*	Name of the database that includes the referencing table (the table with the foreign key). Null if the referencing table is in the current database.
pmrydbname	*varchar(30)*	Name of the database that includes the referenced table (the table with the primary key). Null if the referenced table is in the current database.

Table B-41: Columns in the sysreferences *table*

Indexes

clustered index on *frgndbname, tableid*
unique nonclustered index on *frgndbname, constrid*
nonclustered index on *pmrydbname*, *reftabid*, *indexid*

Referenced by System Procedures

sp_fkeys, sp_helpconstraint

sysremotelogins

(*master* database only)

Description

s*ysremotelogins* contains one row for each remote user who is allowed to execute remote procedure calls on this SQL Server.

Column	Datatype	Description
remoteserverid	*smallint*	Identifies the remote server
remoteusername	*varchar*(30)	User's login name on remote server
suid	*smallint*	Local server user ID
status	*smallint*	Bitmap of options

Table B-42: Columns in the sysremotelogins *table*

Indexes

unique clustered index on *remoteserverid, remoteusername*

Referenced by System Procedures

sp_addremotelogin, sp_checknames, sp_checkreswords, sp_dropremotelogin, sp_dropserver, sp_helpremotelogin, sp_remoteoption

sysroles

(all databases)

Description

sysroles maps server role IDs to local role IDs.

Column	Datatype	Description
id	*smallint*	Server role ID (*srid*)
lrid	*smallint*	Local role ID
type	*smallint*	Unused
status	*smallint*	Unused

Table B-43: Columns in the sysroles *table*

Indexes

Unique clustered index on *lrid*

Referenced by System Procedures

None

syssegments

(all databases)

Description

syssegments contains one row for each segment (named collection of disk pieces). The default entries are: segment 0 (*system*) for system tables; segment 2 *(logsegment*) for the transaction log; and segment 1 (*default*) for other objects. Each database has an entry in *sysusages* contain these segments in its maps.

Column	Datatype	Description
segment	*smallint*	Segment number
name	*sysname*	Segment name
status	*int null*	Indicates which segment is default segment

Table B-44: Columns in the syssegments *table*

Indexes

none

Referenced by System Procedures

sp_addsegment, sp_addthreshold, sp_checknames, sp_checkreswords, sp_dropsegment, sp_dropthreshold, sp_dropuser, sp_extendsegment, sp_helpdb, sp_helpindex, sp_helpsegment, sp_helpthreshold, sp_modifythreshold, sp_placeobject

sysservers

(*master* database only)

Description

sysservers contains one row for each remote SQL Server, Backup Server, or Open Server on which this SQL Server can execute remote procedure calls.

Column	Datatype	Description
srvid	*smallint*	ID number (for local use only) of the remote server
srvstatus	*smallint*	Bitmap of options
srvname	*varchar(30)*	Server name
srvnetname	*varchar(32)*	Interfaces file name for the server

Table B-45: Columns in the sysservers *table*

Indexes

unique clustered index on *srvid*
unique nonclustered index on *srvname*

Referenced by System Procedures

sp_addremotelogin, sp_addserver, sp_checknames, sp_checkreswords, sp_configure, sp_dropremotelogin, sp_dropserver, sp_helpremotelogin, sp_helpserver, sp_remoteoption, sp_serveroption

syssrvroles

(*master* database only)

Description

syssrvroles contains a row for each server-wide role.

Column	Datatype	Description
srid	*smallint*	Server role ID
name	*varchar*	Name of the role

Table B-46: Columns in the syssrvroles *table*

Indexes

Unique clustered index on *srid*

Referenced by System Procedures

sp_adduser, sp_changegroup, sp_displaylogin, sp_dropgroup, sp_helpgroup, sp_role

systhresholds

(all databases)

Description

systhresholds contains one row for each threshold defined for the database.

Column	Datatype	Description
segment	*smallint*	Segment number for which free space is being monitored
free_space	*int*	Size of threshold, in 2K pages (4K for Stratus)
status	*smallint*	Bit 1 equals 1 for the logsegment's last-chance threshold, 0 for all other thresholds
proc_name	*varchar(255)*	Name of the procedure that is executed when the number of unused pages on *segment* falls below *free_space*.
suid	*smallint*	The server user ID of the user who added the threshold or modified it most recently
currauth	*varbinary*(255)	A bit mask that indicates which roles were active for *suid* at the time the threshold was added or most recently modified. When the threshold is crossed, *proc_name* executes with this set of roles, less any that have been deactivated since the threshold was added or last modified.

Table B-47: Columns in the systhresholds *table*

Indexes

Unique clustered index on *segment, free_space*

Referenced by System Procedures

sp_addthreshold, sp_dropsegment, sp_dropthreshold, sp_dropuser, sp_helpthreshold, sp_modifythreshold

systypes

(all databases)

Description

systypes contains one row for each system-supplied and user-defined datatype. Domains (defined by rules) and defaults are given, if they exist.

The rows that describe system-supplied datatypes cannot be altered.

Column	Datatype	Description
uid	*smallint*	User ID of datatype creator
usertype	*smallint*	User type ID
variable	*bit*	1 if datatype is variable length; 0 otherwise
allownulls	*bit*	Indicates whether nulls are allowed for this datatype
type	*tinyint*	Physical storage datatype
length	*tinyint*	Physical length of datatype
tdefault	*int*	ID of system procedure that generates default for this datatype
domain	*int*	ID of system procedure that contains integrity checks for this datatype
name	*sysname*	Datatype name
printfmt	*varchar(255)*	Reserved
prec	*tinyint*	Number of significant digits
scale	*tinyint*	Number of digits to the right of the decimal point
ident	*tinyint*	1 if column has the IDENTITY property, 0 if not
hierarchy	*tinyint*	Precedence of the datatype in mixed mode arithmetic

Table B-48: Columns in the systypes *table*

The listing that follows includes the system-supplied datatype *name*, *hierarchy, type* (not necessarily unique), and *usertype* (unique). The datatypes are ordered by *hierarchy*. In mixed mode arithmetic, the datatype with the lowest *hierarchy* takes precedence:

name	hierarchy	type	usertype
floatn	1	109	14
float	2	62	8
datetimn	3	111	15
datetime	4	61	12
real	5	59	23
numericn	6	108	28
numeric	7	63	10
decimaln	8	106	27
decimal	9	55	26
moneyn	10	110	17
money	11	60	11
smallmoney	12	122	21
smalldatetime	13	58	22
intn	14	38	13
int	15	56	7
smallint	16	52	6
tinyint	17	48	5
bit	18	50	16
varchar	19	39	2
sysname	19	39	18
nvarchar	19	39	25
char	20	47	1
nchar	20	47	24
varbinary	21	37	4
timestamp	21	37	80
binary	22	45	3
text	23	35	19
image	24	34	20

Table B-49: Datatype names, hierarchy, types, and usertypes

Indexes

unique clustered index on *name*
unique nonclustered index on *usertype*

Referenced by System Procedures

sp_addtype, sp_bindefault, sp_bindrule, sp_checknames, sp_checkreswords, sp_columns, sp_datatype_info, sp_droptype, sp_dropuser, sp_help, sp_rename, sp_special_columns, sp_sproc_columns,sp_unbindefault, sp_unbindrule

sysusages

(*master* database only)

Description

sysusages contains one row for each disk allocation piece assigned to a database. Each database contains a specified number of database (logical) page numbers. Each disk piece includes the segments on the SQL Server distribution tape, segments 0 and 1.

The **create database** command checks *sysdevices* and *sysusages* to find available disk allocation pieces. One or more contiguous disk allocation pieces is assigned to the database, and the mapping is recorded in *sysusages*.

Column	Datatype	Description
dbid	*smallint*	Database ID
segmap	*int*	Bit map of possible segment assignments
lstart	*int*	First database (logical) page number
size	*int*	Number of contiguous database (logical) pages
vstart	*int*	Starting virtual page number
pad	*smallint*	Unused
unreservedpgs	*int*	Free space not part of an allocated extent

Table B-50: Columns in the sysusages *table*

Indexes

unique clustered index on *dbid, lstar*t
unique nonclustered index on *vstart*

Referenced by System Procedures

sp_addsegment, sp_addthreshold, sp_databases, sp_dropdevice, sp_dropsegment, sp_extendsegment, sp_helpdb, sp_helplog, sp_helpsegment, sp_logdevice, sp_modifythreshold, sp_spaceused

sysusermessages

(all databases)

Description

sysusermessages contains one row for each user-defined message that can be returned by SQL Server.

Column	Datatype	Description
error	*int*	Unique error number. Must be 20000 or above.
uid	*smallint*	User ID of the message creator
description	*varchar(255)*	User-defined message with optional place holders for parameters
langid	*smallint*	Language ID for this message; null for us_english

Table B-51: Columns in the sysusermessages *table*

Indexes

clustered index on *error*
unique nonclustered index on *error*, *langid*

Referenced by System Procedures

sp_addmessage, sp_bindmsg, sp_dropmessage, sp_getmessage, sp_helpconstraint

sysusers

(all databases)

Description

sysusers contains one row for each user allowed in the database, and one row for each group or role.

On the SQL Server distribution tape, *master..sysusers* contains some initial users: "dbo," whose *suid* is 1 and *uid* is 1; "guest," whose *suid* is -1 and *uid* is 2; and "public," whose *suid* is -2 and *uid* is 0. In addition, each role (**sa_role**, **sso_role**, and so on) is listed in *sysusers,* because SQL Server treats roles much like groups.

The user *guest* provides a mechanism for giving users not explicitly listed in *sysusers* access to the database with a restricted set of permissions. The "guest" entry in *master* means that any user with an account on SQL Server (that is, with an entry in *syslogins*) can access *master.*

The user "public" refers to all users. The keyword **public** is used with the **grant** and **revoke** commands to signify that permission is being given to or taken away from all users.

Column	Datatype	Description
suid	*smallint*	Server user ID, copied from *syslogins*.
uid	*smallint*	User ID, unique in this database, used for granting and revoking permissions. User ID 1 is "dbo".
gid	*smallint*	Group ID to which this user belongs. If *uid* = *gid,* this entry defines a group. The group "public" has *suid* = -2; all other groups have *suid*= - *gid*.
name	*sysname*	User or group name, unique in this database
environ	*varchar(255)*	Reserved

Table B-52: Columns in the sysusers *table*

Indexes

unique clustered index on *suid*
unique nonclustered index on *name*
unique nonclustered index on *uid*

Referenced by System Procedures

sp_addalias, sp_addgroup, sp_adduser, sp_changedbowner, sp_changegroup, sp_checknames, sp_checkreswords, sp_column_privileges, sp_depends, sp_dropgroup, sp_droptype, sp_dropuser, sp_helpgroup, sp_helprotect, sp_helpuser, sp_indsuspect, sp_stored_procedures, sp_table_privileges, sp_tables

C The *pubs2* Database

This is the sample database *pubs2*. The names of the 11 tables are *publishers*, *authors*, *titles*, *titleauthor*, *sales*, *salesdetail*, *stores*, *discounts*, *roysched*, *au_pix*, and *blurbs*.

The header for each column lists its datatype (including the user-defined datatypes) and its null / not null status. Defaults, rules, triggers, and indexes are noted where they apply.

Tables in the *pubs2* Database

publishers			
pub_id char(4) not null pub_idrule[a] clust, uniq	pub_name varchar(40) null	city varchar(20) null	state char(2) null
0736	New Age Books	Boston	MA
0877	Binnet & Hardley	Washington	DC
1389	Algodata Infosystems	Berkeley	CA

a. The pub_id rule states that the data must be 1389, 0736, 0877, 1622, or 1756, or must match the pattern 99[0-9][0-9].

authors								
au_id	au_lname	au_fname	phone	address	city	state	country	postalcode
id	varchar(40)	varchar(20)	char(12)	varchar(12)	varchar(20)	char(2)	varchar(12)	char(10)
not null	not null	not null	not null	null	null	null	null	null
			UNKNOWN[a]					
clust, uniq	nonclust							
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	USA	94025
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	USA	94618
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	USA	94705
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	USA	95128
274-80-9391	Straight	Dick	415 834-2919	5420 College Av.	Oakland	CA	USA	94609
341-22-1782	Smith	Meander	913 843-0462	10 Mississippi Dr.	Lawrence	KS	USA	66044
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	USA	94705
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	USA	94301
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA	USA	95428
486-29-1786	Locksley	Chastity	415 585-4620	18 Broadway Av.	San Francisco	CA	USA	94130
527-72-3246	Greene	Morningstar	615 297-2723	22 Graybar House Rd.	Nashville	TN	USA	37215
648-92-1872	Blotchet-Halls	Reginald	503 745-6402	55 Hillsdale Bl.	Corvallis	OR	USA	97330
672-71-3249	Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA	USA	94595
712-45-1867	del Castillo	Innes	615 996-8275	2286 Cram Pl. #86	Ann Arbor	MI	USA	48105
722-51-5454	DeFrance	Michel	219 547-9982	3 Balding Pl.	Gary	IN	USA	46403
724-08-9931	Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA	USA	94609
724-80-9391	MacFeather	Stearns	415 354-7128	44 Upland Hts.	Oakland	CA	USA	94612
756-30-7391	Karsen	Livia	415 534-9219	5720 McAuley St.	Oakland	CA	USA	94609
807-91-6654	Panteley	Sylvia	301 946-8853	1956 Arlington Pl.	Rockville	MD	USA	20853
846-92-7186	Hunter	Sheryl	415 836-7128	3410 Blonde St.	Palo Alto	CA	USA	94301
893-72-1158	McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA	USA	95688
899-46-2035	Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT	USA	84152
998-72-3567	Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT	USA	84152

a. The default *UNKNOWN* is inserted if no data is entered.

titles									
title_id tid not null deltitle[3] clust, uniq	title varchar(80) not null nonclust	type char(12) not null UNDECIDED[1]	pub_id char(4) null	price money null	advance money null	total_sales int null	notes varchar(200) null	pubdate datetime not null getdate()[2]	contract bit not null
BU1032	The Busy Executive's Database Guide	business	1389	19.99	5000.00	4095	An overview of available database systems with emphasis on common business applications. Illustrated.	Jun 6, 1986	1
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	1389	11.95	5000.00	3876	Helpful hints on how to use your electronic resources to the best advantage.	Jun 9, 1988	1
BU2075	You Can Combat Computer Stress!	business	0736	2.99	10125.00	18722	The latest medical and psychological techniques for living with the electronic office. Easy-to-understand explanations.	Jun 30, 1985	1
BU7832	Straight Talk About Computers	business	1389	19.99	5000.00	4095	Annotated analysis of what computers can do for you: a no-hype guide for the critical user.	Jun 22, 1987	1
MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19.99	0.00	2032	Favorite recipes for quick, easy, and elegant meals, tried and tested by people who never have time to eat, let alone cook.	Jun 9, 1989	1
MC3021	The Gourmet Microwave	mod_cook	0877	2.99	15000.00	22246	Traditional French gourmet recipes adapted for modern microwave cooking.	Jun 18, 1985	1
PC1035	But Is It User Friendly?	popular_comp	1389	22.95	7000.00	8780	A survey of software for the naive user, focusing on the 'friendliness' of each.	Jun 30, 1986	1
MC3026	The Psychology of Computer Cooking	UNDECIDED	0877	NULL	NULL	NULL	NULL	Jul 24, 1991	0
PC8888	Secrets of Silicon Valley	popular_comp	1389	20.00	8000.00	4095	Muckraking reporting by two courageous women on the world's largest computer hardware and software manufacturers.	Jun 12, 1987	1

1. The default *UNDECIDED* is inserted if no data is entered in the column.
2. The *getdate* function inserts the current date as the default if no data is entered in the column.
3. The *deltitle* trigger prohibits deleting a title if the *title_id* is listed in the *sales* table.

titles									
title_id tid not null deltitle[3] clust, uniq	title varchar(80) not null nonclust	type char(12) not null UNDECIDED[1]	pub_id char(4) null	price money null	advance money null	total_sales int null	notes varchar(200) null	pubdate datetime not null getdate()[2]	contract bit not null
PC9999	Net Etiquette	popular_comp	1389	NULL	NULL	NULL	A must-read for computer conferencing debutantes!	Jul 24, 1991	0
PS1372	Computer Phobic and Non-Phobic Individuals: Behavior Variations	psychology	0877	21.59	7000.00	375	A must for the specialist, this book examines the difference between those who hate and fear computers and those who think they are swell.	Oct 21,1990	1
PS2091	Is Anger the Enemy?	psychology	0736	10.95	2275.00	2045	Carefully researched study of the effects of strong emotions on the body. Metabolic charts included.	Jun 15, 1989	1
PS2106	Life Without Fear	psychology	0736	7.00	6000.00	111	New exercise, meditation, and nutritional techniques that can reduce the shock of daily interactions. Popular audience. Sample menus included, exercise video available separately.	Oct 5, 1990	1
PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	0736	19.99	2000.00	4072	What happens when the data runs dry? Searching evaluations of information-shortage effects on heavy users.	Jun 12, 1988	1
PS7777	Emotional Security: A New Algorithm	psychology	0736	7.99	4000.00	3336	Protecting yourself and your loved ones from undue emotional stress in the modern world. Use of computer and nutritional aids emphasized.	Jun 12, 1988	1
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	0877	20.95	7000.00	375	Profusely illustrated in color, this makes a wonderful gift book for a cuisine-oriented friend.	Oct 21, 1990	1
TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	0877	11.95	4000.00	15096	More anecdotes from the Queen's favorite cook describing life among English royalty. Recipes, techniques, tender vignettes.	Jun 12, 1985	1

1. The default *UNDECIDED* is inserted if no data is entered in the column.
2. The *getdate* function inserts the current date as the default if no data is entered in the column.
3. The *deltitle* trigger prohibits deleting a title if the *title_id* is listed in the *sales* table.

titles									
title_id tid not null deltitle[3] clust, uniq	title varchar(80) not null nonclust	type char(12) not null UNDECIDED[1]	pub_id char(4) null	price money null	advance money null	total_sales int null	notes varchar(200) null	pubdate datetime not null getdate()[2]	contract bit not null
TC7777	Sushi, Anyone?	trad_cook	0877	14.99	8000.00	4095	Detailed instructions on improving your position in life by learning how to make authentic Japanese sushi in your spare time. 5-10% increase in number of friends per recipe reported from beta test.	Jun 12, 1987	1

1. The default *UNDECIDED* is inserted if no data is entered in the column.
2. The *getdate* function inserts the current date as the default if no data is entered in the column.
3. The *deltitle* trigger prohibits deleting a title if the *title_id* is listed in the *sales* table.

titleauthor			
au_id id not null nonclust	title_id tid not null nonclust	au_ord tinyint null	royaltyper int null
uniq, clust, composite			
172-32-1176	PS3333	1	100
213-46-8915	BU1032	2	40
213-46-8915	BU2075	1	100
238-95-7766	PC1035	1	100
267-41-2394	BU1111	2	40
267-41-2394	TC7777	2	30
274-80-9391	BU7832	1	100
409-56-7008	BU1032	1	60
427-17-2319	PC8888	1	50
472-27-2349	TC7777	3	30
486-29-1786	PC9999	1	100
486-29-1786	PS7777	1	100
648-92-1872	TC4203	1	100
672-71-3249	TC7777	1	40
712-45-1867	MC2222	1	100
722-51-5454	MC3021	1	75
724-80-9391	BU1111	1	60
724-80-9391	PS1372	2	25
756-30-7391	PS1372	1	75
807-91-6654	TC3218	1	100
846-92-7186	PC8888	2	50
899-46-2035	MC3021	2	25
899-46-2035	PS2091	2	50
998-72-3567	PS2091	1	50
998-72-3567	PS2106	1	100

au_pix					
au_id id not null	pic image null	format_type char(11) null	bytesize int null	pixwidth_hor char(14) null	pixwidth_vert char(14) null
409-56-7008	0x0000...	PICT	30220	626	635
486-29-1786	0x59a6...	Sunraster	27931	647	640
648-92-1872	0x59a6...	Sunraster	36974	647	640
672-71-3249	0x000a...	PICT	13487	654	639
899-46-2035	0x4949...	TIF	52023	648	641
998-72-3567	0x4949...	TIF	52336	653	637

The *pic* column contains binary data, which is not reproduced in this table in its entirety. The pictures represented by this data are shown on the next page. Since the *image* data (six pictures, two each in PICT, TIF, and Sun raster file formats) is quite large, you should run the *installpix2* script **only** if you want to use or test the *image* datatype. The *image* data is supplied to show how SYBASE stores *image* data. Sybase does not supply any tools for displaying *image* data: you must use the appropriate screen graphics tools in order to display the images once you have extracted them from the database.

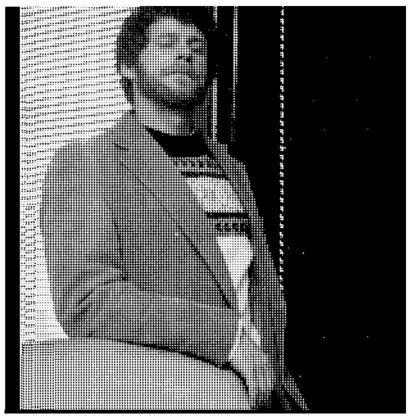
Authors' Portraits from the *au_pix* Table

Akiko Yokomoto 672-71-3249

Chastity Locksley 486-29-1786

Anne Ringer 899-46-2035

Albert Ringer 998-72-3567


Bennet Abraham 409-56-7008

Reginald Blotchet-Halls 648-92-1872

salesdetail				
stor_id char(4) not null	ord_num varchar(20) not null	title_id tid not null title_idrule	qty smallint not null	discount float not null
nonclust		nonclust		
7896	234518	TC3218	75	40.000000
7896	234518	TC7777	75	40.000000
7131	Asoap432	TC3218	50	40.000000
7131	Asoap432	TC7777	80	40.000000
5023	XS-135-DER-432-8J2	TC3218	85	40.000000
8042	91-A-7	PS3333	90	45.000000
8042	91-A-7	TC3218	40	45.000000
8042	91-A-7	PS2106	30	45.000000
8042	91-V-7	PS2106	50	45.000000
8042	55-V-7	PS2106	31	45.000000
8042	91-A-7	MC3021	69	45.000000
5023	BS-345-DSE-860-1F2	PC1035	1000	46.700000
5023	AX-532-FED-452-2Z7	BU2075	500	46.700000
5023	AX-532-FED-452-2Z7	BU1032	200	46.700000
5023	AX-532-FED-452-2Z7	BU7832	150	46.700000
5023	AX-532-FED-452-2Z7	PS7777	125	46.700000
5023	NF-123-ADS-642-9G3	TC7777	1000	46.700000
5023	NF-123-ADS-642-9G3	BU1032	1000	46.700000
5023	NF-123-ADS-642-9G3	PC1035	750	46.700000
7131	Fsoap867	BU1032	200	46.700000
7066	BA52498	BU7832	100	46.700000
7066	BA71224	PS7777	200	46.700000
7066	BA71224	PC1035	300	46.700000
7066	BA71224	TC7777	350	46.700000
5023	ZD-123-DFG-752-9G8	PS2091	1000	46.700000
7067	NB-3.142	PS2091	200	46.700000
7067	NB-3.142	PS7777	250	46.700000
7067	NB-3.142	PS3333	345	46.700000
7067	NB-3.142	BU7832	360	46.700000
5023	XS-135-DER-432-8J2	PS2091	845	46.700000
5023	XS-135-DER-432-8J2	PS7777	581	46.700000
5023	ZZ-999-ZZZ-999-0A0	PS1372	375	46.700000
7067	NB-3.142	BU1111	175	46.700000
5023	XS-135-DER-432-8J2	BU7832	885	46.700000
5023	ZD-123-DFG-752-9G8	BU7832	900	46.700000
5023	AX-532-FED-452-2Z7	TC4203	550	46.700000
7131	Fsoap867	TC4203	350	46.700000
7896	234518	TC4203	275	46.700000
7066	BA71224	TC4203	500	46.700000

salesdetail				
stor_id char(4) not null	ord_num varchar(20) not null	title_id tid not null title_idrule	qty smallint not null	discount float not null
nonclust		nonclust		
7067	NB-3.142	TC4203	512	46.700000
7131	Fsoap867	MC3021	400	46.700000
5023	AX-532-FED-452-2Z7	PC8888	105	46.700000
5023	NF-123-ADS-642-9G3	PC8888	300	46.700000
7066	BA71224	PC8888	350	46.700000
7067	NB-3.142	PC8888	335	46.700000
7131	Asoap432	BU1111	500	46.700000
7896	234518	BU1111	340	46.700000
5023	AX-532-FED-452-2Z7	BU1111	370	46.700000
5023	ZD-123-DFG-752-9G8	PS3333	750	46.700000
8042	13-J-9	BU7832	300	51.700000
8042	13-E-7	BU2075	150	51.700000
8042	13-E-7	BU1032	300	51.700000
8042	13-E-7	PC1035	400	51.700000
8042	91-A-7	PS7777	180	51.700000
8042	13-J-9	TC4203	250	51.700000
8042	13-E-7	TC4203	226	51.700000
8042	13-E-7	MC3021	400	51.700000
8042	91-V-7	BU1111	390	51.700000
5023	AB-872-DEF-732-2Z1	MC3021	5000	50.000000
5023	NF-123-ADS-642-9G3	PC8888	2000	50.000000
5023	NF-123-ADS-642-9G3	BU2075	2000	50.000000
5023	GH-542-NAD-713-9F9	PC1035	2000	50.000000
5023	ZA-000-ASD-324-4D1	PC1035	2000	50.000000
5023	ZA-000-ASD-324-4D1	PS7777	1500	50.000000
5023	ZD-123-DFG-752-9G8	BU2075	3000	50.000000
5023	ZD-123-DFG-752-9G8	TC7777	1500	50.000000
5023	ZS-645-CAT-415-1B2	BU2075	3000	50.000000
5023	ZS-645-CAT-415-1B2	BU2075	3000	50.000000
5023	XS-135-DER-432-8J2	PS3333	2687	50.000000
5023	XS-135-DER-432-8J2	TC7777	1090	50.000000
5023	XS-135-DER-432-8J2	PC1035	2138	50.000000
5023	ZZ-999-ZZZ-999-0A0	MC2222	2032	50.000000
5023	ZZ-999-ZZZ-999-0A0	BU1111	1001	50.000000
5023	ZA-000-ASD-324-4D1	BU1111	1100	50.000000
5023	NF-123-ADS-642-9G3	BU7832	1400	50.000000
5023	BS-345-DSE-860-1F2	TC4203	2700	50.000000
5023	GH-542-NAD-713-9F9	TC4203	2500	50.000000
5023	NF-123-ADS-642-9G3	TC4203	3500	50.000000

salesdetail				
stor_id char(4) not null	ord_num varchar(20) not null	title_id tid not null title_idrule	qty smallint not null	discount float not null
nonclust		nonclust		
5023	BS-345-DSE-860-1F2	MC3021	4500	50.000000
5023	AX-532-FED-452-2Z7	MC3021	1600	50.000000
5023	NF-123-ADS-642-9G3	MC3021	2550	50.000000
5023	ZA-000-ASD-324-4D1	MC3021	3000	50.000000
5023	ZS-645-CAT-415-1B2	MC3021	3200	50.000000
5023	BS-345-DSE-860-1F2	BU2075	2200	50.000000
5023	GH-542-NAD-713-9F9	BU1032	1500	50.000000
5023	ZZ-999-ZZZ-999-0A0	PC8888	1005	50.000000
7896	124152	BU2075	42	50.500000
7896	124152	PC1035	25	50.500000
7131	Asoap132	BU2075	35	50.500000
7067	NB-1.142	PC1035	34	50.500000
7067	NB-1.142	TC4203	53	50.500000
8042	12-F-9	BU2075	30	55.500000
8042	12-F-9	BU1032	94	55.500000
7066	BA27618	BU2075	200	57.200000
7896	124152	TC4203	350	57.200000
7066	BA27618	TC4203	230	57.200000
7066	BA27618	MC3021	200	57.200000
7131	Asoap132	MC3021	137	57.200000
7067	NB-1.142	MC3021	270	57.200000
7067	NB-1.142	BU2075	230	57.200000
7131	Asoap132	BU1032	345	57.200000
7067	NB-1.142	BU1032	136	57.200000
8042	12-F-9	TC4203	300	62.200000
8042	12-F-9	MC3021	270	62.200000
8042	12-F-9	PC1035	133	62.200000
5023	AB-123-DEF-425-1Z3	TC4203	2500	60.500000
5023	AB-123-DEF-425-1Z3	BU2075	4000	60.500000
6380	342157	BU2075	200	57.200000
6380	342157	MC3021	250	57.200000
6380	356921	PS3333	200	46.700000
6380	356921	PS7777	500	46.700000
6380	356921	TC3218	125	46.700000
6380	234518	BU2075	135	46.700000
6380	234518	BU1032	320	46.700000
6380	234518	TC4203	300	46.700000
6380	234518	MC3021	400	46.700000

sales		
stor_id char(4) not null	ord_num varchar(20) not null	date datetime not null
clust, uniq	clust, uniq	
5023	AB-123-DEF-425-1Z3	Oct 31 1985
5023	AB-872-DEF-732-2Z1	Nov 6 1985
5023	AX-532-FED-452-2Z7	Dec 1 1990
5023	BS-345-DSE-860-1F2	Dec 12 1986
5023	GH-542-NAD-713-9F9	Mar 15 1987
5023	NF-123-ADS-642-9G3	Jul 18 1987
5023	XS-135-DER-432-8J2	Mar 21 1991
5023	ZA-000-ASD-324-4D1	Jul 27 1988
5023	ZD-123-DFG-752-9G8	Mar 21 1991
5023	ZS-645-CAT-415-1B2	Mar 21 1991
5023	ZZ-999-ZZZ-999-0A0	Mar 21 1991
6380	234518	Sep 30 1987
6380	342157	Dec 13 1985
6380	356921	Feb 17 1991
7066	BA27618	Oct 12 1985
7066	BA52498	Oct 27 1987
7066	BA71224	Aug 5 1988
7067	NB-1.142	Jan 2 1987
7067	NB-3.142	Jun 13 1990
7131	Asoap132	Nov 16 1986
7131	Asoap432	Dec 20 1990
7131	Fsoap867	Sep 8 1987
7896	124152	Aug 14 1986
7896	234518	Feb 14 1991
8042	12-F-9	Jul 13 1986
8042	13-E-7	May 23 1989
8042	13-J-9	Jan 13 1988
8042	55-V-7	Mar 20 1991
8042	91-A-7	Mar 20 1991
8042	91-V-7	Mar 20 1991

stores							
stor_id char(4) not null	stor_name varchar(40) null	stor_address varchar(40) null	city varchar(20) null	state char(2) null	country varchar(12) null	postalcode char(10) null	payterms varchar(12) null
7066	Barnum's	567 Pasadena Ave.	Tustin	CA	USA	92789	Net 30
7067	News & Brews	577 First St.	Los Gatos	CA	USA	96745	Net 30
7131	Doc-U-Mat: Quality Laundry and Books	24-A Avrogado Way	Remulade	WA	USA	98014	Net 60
8042	Bookbeat	679 Carson St.	Portland	OR	USA	89076	Net 30
6380	Eric the Read Books	788 Catamaugus Ave.	Seattle	WA	USA	98056	Net 60
7896	Fricative Bookshop	89 Madison St.	Fremont	CA	USA	90019	Net 60
5023	Thoreau Reading Discount Chain	20435 Walden Expressway	Concord	MA	USA	01776	Net 60

discounts				
discounttype varchar(40) not null	stor_id char(4) null	lowqty smallint null	highqty smallint null	discount float not null
Initial Customer				10.5
Volume Discount		100	1000	6.7
Huge Volume Discount		1001		10
Customer Discount	8042			5

roysched			
title_id tid not null nonclust	lorange int null	hirange int null	royalty int null
BU1032	0	5000	10
BU1032	5001	50000	12
PC1035	0	2000	10
PC1035	2001	3000	12
PC1035	3001	4000	14
PC1035	4001	10000	16
PC1035	10001	50000	18
BU2075	0	1000	10
BU2075	1001	3000	12
BU2075	3001	5000	14
BU2075	5001	7000	16
BU2075	7001	10000	18
BU2075	10001	12000	20
BU2075	12001	14000	22
BU2075	14001	50000	24
PS2091	0	1000	10
PS2091	1001	5000	12
PS2091	5001	10000	14
PS2091	10001	50000	16
PS2106	0	2000	10
PS2106	2001	5000	12
PS2106	5001	10000	14
PS2106	10001	50000	16
MC3021	0	1000	10
MC3021	1001	2000	12
MC3021	2001	4000	14
MC3021	4001	6000	16
MC3021	6001	8000	18
MC3021	8001	10000	20
MC3021	10001	12000	22

roysched			
title_id tid not null nonclust	lorange int null	hirange int null	royalty int null
MC3021	12001	50000	24
TC3218	0	2000	10
TC3218	2001	4000	12
TC3218	4001	6000	14
TC3218	6001	8000	16
TC3218	8001	10000	18
TC3218	10001	12000	20
TC3218	12001	14000	22
TC3218	14001	50000	24
PC8888	0	5000	10
PC8888	5001	10000	12
PC8888	10001	15000	14
PC8888	15001	50000	16
PS7777	0	5000	10
PS7777	5001	50000	12
PS3333	0	5000	10
PS3333	5001	10000	12
PS3333	10001	15000	14
PS3333	15001	50000	16
BU1111	0	4000	10
BU1111	4001	8000	12
BU1111	8001	10000	14
BU1111	12001	16000	16
BU1111	16001	20000	18
BU1111	20001	24000	20
BU1111	24001	28000	22
BU1111	28001	50000	24
MC2222	0	2000	10
MC2222	2001	4000	12
MC2222	4001	8000	14

roysched			
title_id tid not null nonclust	lorange int null	hirange int null	royalty int null
MC2222	8001	12000	16
MC2222	8001	12000	16
MC2222	12001	20000	18
MC2222	20001	50000	20
TC7777	0	5000	10
TC7777	5001	15000	12
TC7777	15001	50000	14
TC4203	0	2000	10
TC4203	2001	8000	12
TC4203	8001	16000	14
TC4203	16001	24000	16
TC4203	24001	32000	18
TC4203	32001	40000	20
TC4203	40001	50000	22
BU7832	0	5000	10
BU7832	5001	10000	12
BU7832	10001	15000	14
BU7832	15001	20000	16
BU7832	20001	25000	18
BU7832	25001	30000	20
BU7832	30001	35000	22
BU7832	35001	50000	24
PS1372	0	10000	10
PS1372	10001	20000	12
PS1372	20001	30000	14
PS1372	30001	40000	16
PS1372	40001	50000	18

blurbs	
au_id id not null	copy text null
486-29-1786	If Chastity Locksley didn’t exist, this troubled world would have created her! Not only did she master the mystic secrets of inner strength to conquer adversity when she encountered it in life, but, after “reinventing herself”, as she says, by writing “Emotional Security: A New Algorithm” following the devastating loss of her cat Old Algorithm, she also founded Publish or Perish, the page-by-page, day-by-day, write-yourself-to-wellness encounter workshops franchise empire, the better to share her inspiring discoveries with us all. Her “Net Etiquette,” a brilliant social treatise in its own right and a fabulous pun, is the only civilized alternative to the gross etiquette often practiced on the public networks.
648-92-1872	A chef’s chef and a raconteur’s raconteur, Reginald Blotchet-Halls calls London his second home. “Th’ palace kitchen’s me first ‘ome, act’lly!” Blotchet-Halls’ astounding ability to delight our palates with palace delights is matched only by his equal skill in satisfying our perpetual hunger for delicious back-stairs gossip by serving up tidbits and entrees literally fit for a king!
998-72-3567	Albert Ringer was born in a trunk to circus parents, but another kind of circus trunk played a more important role in his life years later. He grew up as an itinerant wrestler and roustabout in the reknowned Ringer Brothers and Betty and Bernie’s Circus. Once known in the literary world only as Anne Ringer’s wrestling brother, he became a writer while recuperating from a near-fatal injury received during a charity benefit bout with a gorilla. “Slingshotting” himself from the ring ropes, Albert flew over the gorilla’s head and would have landed head first on the concrete. He was saved from certain death by Nana, an elephant he had befriended as a child, who caught him in her trunk. Nana held him so tightly that three ribs cracked and he turned blue from lack of oxygen. “I was delirious. I had an out-of-body experience! My whole life passed before me eyes. I promised myself ‘If I get through this, I’ll use my remaining time to share what I learned out there.’ I owe it all to Nana!”
899-46-2035	Anne Ringer ran away from the circus as a child. A university creative writing professor and her family took Anne in and raised her as one of their own. In this warm and television-less setting she learned to appreciate the great classics of literature. The stream of aspiring and accomplished writers that flowed constantly through the house confirmed her repudiation of the circus family she’d been born into: “Barbarians!” The steadily growing recognition of her literary work was, to her, vindication. When her brother’s brush with death brought them together after many years, she took advantage of life’s crazy chance thing and broke the wall of anger that she had constructed to separate them. Together they wrote, “Is Anger the Enemy?” an even greater blockbuster than her other collaborative work, with Michel DeFrance, “The Gourmet Microwave.”
672-71-3249	They asked me to write about myself and my book, so here goes: I started a restaurant called “de Gustibus” with two of my friends. We named it that because you really can’t discuss taste. We’re very popular with young business types because we’re young business types ourselves. Whenever we tried to go out to eat in a group we always got into these long tiresome negotiations: “I just ate Italian,” or “I ate Greek yesterday,” or “I NEVER eat anything that’s not organic!” Inefficient. Not what business needs today. So, it came to us that we needed a restaurant we could all go to every day and not eat the same thing twice in a row maybe for a year! We thought, “Hey, why make people choose one kind of restaurant over another, when what they really want is a different kind of food?” At de Gustibus you can eat Italian, Chinese, Japanese, Greek, Russian, Tasmanian, Iranian, and on and on all at the same time. You never have to choose. You can even mix and match! We just pooled our recipes, opened the doors, and never looked back. We’re a big hit, what can I say? My recipes in “Sushi, Anyone?” are used at de Gustibus. They satisfy crowds for us every day. They will work for you, too. Period!
409-56-7008	Bennet was the classic too-busy executive. After discovering computer databases he now has the time to run several successful businesses and sit on three major corporate boards. Bennet also donates time to community service organizations. Miraculously, he also finds time to write and market executive-oriented in-depth computer hardware and software reviews. “I’m hyperkinetic, so being dynamic and fast-moving is a piece of cake. But being organized isn’t easy for me or for anyone I know. There’s just one word for that: ‘databases!’ Databases can cure you or kill you. If you get the right one, you can be like me. If you get the wrong one, watch out. Read my book!”

Primary and Foreign Keys in *pubs2*

Primary Keys	
Table	Primary Key
titles	title_id
titleauthor	au_id + title_id
authors	au_id
publishers	pub_id
roysched	title_id
sales	stor_id + ord_num
salesdetail	stor_id + ord_num
stores	stor_id
discounts	discounttype + stor_id
au_pix	au_id
blurbs	au_id

Foreign Keys		
Table	Foreign Key	Primary Key Table
titleauthor	title_id au_id	titles authors
roysched	title_id	titles
sales	title_id stor_id	titles stores
salesdetail	title_id stor_id, ord_num	titles sales
titles	pub_id	publishers
discounts	stor_id	stores
au_pix	au_id	authors
blurbs	au_id	authors

Other Objects in *pubs2*

Rules

pub_idrule

```
create rule pub_idrule
as @pub_id in (“1389”, “0736”, “0877”, “1622”,
“1756”)
or @pub_id like “99[0-9][0-9]”
```

title_idrule

```
create rule title_idrule
as
@title_id like “BU[0-9][0-9][0-9][0-9]” or
@title_id like “[MT]C[0-9][0-9][0-9][0-9]” or
@title_id like “P[SC][0-9][0-9][0-9][0-9]” or
@title_id like “[A-Z][A-Z]xxxx” or
@title_id like “[A-Z][A-Z]yyyy”
/*valid values: BU, MC, TC, PS, PC + 4 digits or
**any two uppercase letters followed by x’s or y’s
*/
```

Defaults

typedflt

```
create default typedflt as “UNDECIDED”
```

datedflt

```
create default datedflt as getdate()
```

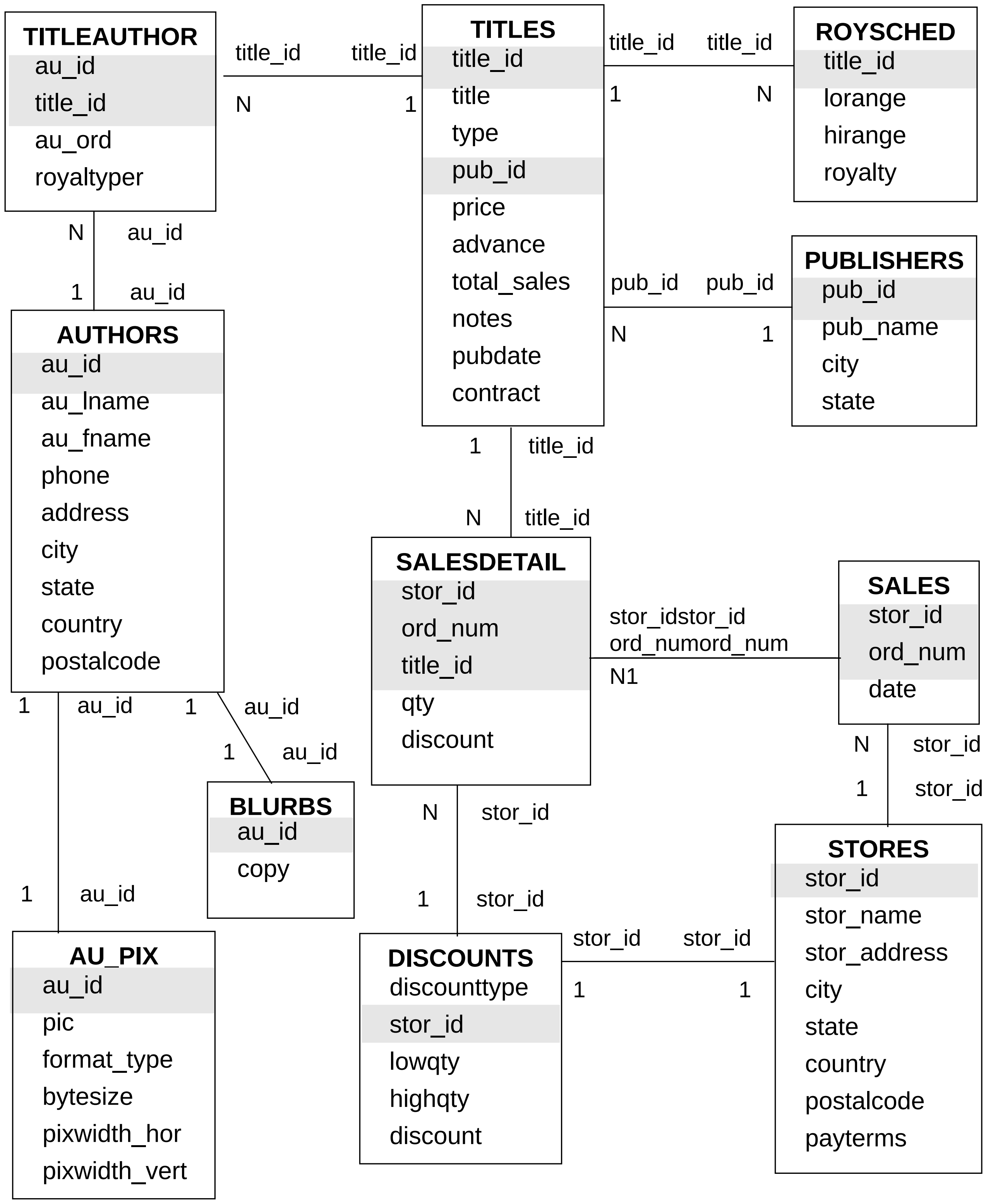
phonedflt

```
create default phonedflt as “UNKNOWN”
```

View

```
create view titleview
as
select title, au_ord, au_lname,
price, total_sales, pub_id
from authors, titles, titleauthor
where authors.au_id = titleauthor.au_id
and titles.title_id = titleauthor.title_id
```

Diagram of the *pubs2* Database



Index

Index

The index is divided into three sections:

- Symbols

  Indexes each of the symbols used in SYBASE SQL Server documentation.

- Numbers

  Indexes entries which begin numerically.

- Subjects

  Indexes subjects alphabetically.

Page numbers in **bold** are primary references.