SYBASE®

Reference Manual: Procedures

Adaptive Server® Enterprise

12.5.1

DOCUMENT ID: DC36273-01-1251-03

LAST REVISED: November 2004

Contents

About This Book .... **xi**

CHAPTER 1 **System Procedures** **1**

Introduction to system procedures 1
Permissions on system procedures 2
Executing system procedures 2
Entering parameter values 3
Messages 4
System procedure tables 5
List of system procedures 5
sp_activeroles 15
sp_addalias 16
sp_addauditrecord 17
sp_addaudittable 19
sp_addengine 21
sp_addexeclass 22
sp_addextendedproc 23
sp_addexternlogin 25
sp_addgroup 28
sp_addlanguage 29
sp_addlogin 32
sp_addmessage 35
sp_addobjectdef 37
sp_add_qpgroup 40
sp_addremotelogin 41
sp_add_resource_limit 44
sp_addsegment 49
sp_addserver 51
sp_addthreshold 54
sp_add_time_range 58
sp_addtype 61
sp_addumpdevice 65
sp_adduser 67
sp_altermessage 69

sp_audit 70
sp_autoconnect 76
sp_bindcache 78
sp_bindefault 82
sp_bindexeclass 84
sp_bindmsg 87
sp_bindrule 88
sp_cacheconfig 90
sp_cachestrategy 98
sp_changedbowner 101
sp_changegroup 102
sp_checknames 103
sp_checkreswords 104
sp_checksource 117
sp_chgattribute 119
sp_clearpsexe 123
sp_clearstats 124
sp_client_addr 126
sp_cmp_all_qplans 128
sp_cmp_qplans 130
sp_commonkey 132
sp_companion 134
sp_configure 137
sp_copy_all_qplans 143
sp_copy_qplan 144
sp_countmetadata 145
sp_cursorinfo 147
sp_dbextend 150
sp_dboption 157
sp_dbrecovery_order 165
sp_dbremap 167
sp_defaultloc 168
sp_depends 171
sp_deviceattr 177
sp_diskdefault 179
sp_displayaudit 181
sp_displaylevel 185
sp_displaylogin 186
sp_displayroles 189
sp_dropalias 191
sp_drop_all_qplans 192
sp_dropdevice 193
sp_dropengine 194
sp_dropexeclass 195

sp_dropextendedproc 196
sp_dropexternlogin 197
sp_dropglockpromote 198
sp_dropgroup 199
sp_dropkey 200
sp_droplanguage 202
sp_droplogin 203
sp_dropmessage 204
sp_dropobjectdef 205
sp_drop_qpgroup 207
sp_drop_qplan 208
sp_dropremotelogin 209
sp_drop_resource_limit 210
sp_droprowlockpromote 213
sp_dropsegment 214
sp_dropserver 216
sp_dropthreshold 217
sp_drop_time_range 218
sp_droptype 219
sp_dropuser 220
sp_dumpoptimize 221
sp_engine 226
sp_estspace 229
sp_export_qpgroup 234
sp_extendsegment 235
sp_extengine 236
sp_familylock 237
sp_find_qplan 240
sp_fixindex 242
sp_flushstats 244
sp_forceonline_db 245
sp_forceonline_object 246
sp_forceonline_page 248
sp_foreignkey 250
sp_freedll 252
sp_getmessage 253
sp_grantlogin 254
sp_ha_admin 256
sp_help 257
sp_helpartition 264
sp_helpcache 267
sp_helpconfig 269
sp_helpconstraint 274
sp_helpdb 278

sp_helpdevice 281
sp_helpextendedproc 283
sp_helpexternlogin 284
sp_helpgroup 285
sp_helpindex 286
sp_helpjava 288
sp_helpjoins 290
sp_helpkey 292
sp_helplanguage 294
sp_helplog 295
sp_helpobjectdef 296
sp_help_qpgroup 297
sp_help_qplan 299
sp_helpremotelogin 300
sp_help_resource_limit 301
sp_helprotect 304
sp_helpsegment 308
sp_helpserver 311
sp_helpsort 312
sp_helptext 314
sp_helpthreshold 316
sp_helpuser 317
sp_hidetext 318
sp_import_qpgroup 320
sp_indsuspect 322
sp_ldapadmin 323
sp_listener 326
sp_listsuspect_db 329
sp_listsuspect_object 330
sp_listsuspect_page 331
sp_lock 332
sp_locklogin 336
sp_logdevice 338
sp_loginconfig 340
sp_logininfo 342
sp_logiosize 343
sp_modifylogin 346
sp_modify_resource_limit 349
sp_modify_time_range 352
sp_modifystats 354
sp_modifythreshold 357
sp_monitor 361
sp_monitorconfig 364
sp_object_stats 371

sp_passthru 374
sp_password 376
sp_placeobject 378
sp_plan_dbccdb 380
sp_poolconfig 382
sp_primarykey 387
sp_processmail 388
sp_procxmode 390
sp_recompile 392
sp_remap 393
sp_remoteoption 394
sp_remotesql 396
sp_rename 398
sp_renamedb 400
sp_rename_qpgroup 402
sp_reportstats 403
sp_revokelogin 405
sp_role 406
sp_sendmsg 407
sp_serveroption 409
sp_setlangalias 413
sp_setpglockpromote 414
sp_setpsexe 416
sp_set_qplan 417
sp_setrowlockpromote 418
sp_setsuspect_granularity 420
sp_setsuspect_threshold 423
sp_showcontrolinfo 424
sp_showexeclass 426
sp_showplan 427
sp_showpsexe 429
sp_spaceused 430
sp_ssladmin 433
sp_syntax 435
sp_sysmon 437
sp_tempdb 440
sp_thresholdaction 448
sp_transactions 450
sp_unbindcache 457
sp_unbindcache_all 459
sp_unbindefault 460
sp_unbindexeclass 461
sp_unbindmsg 463
sp_unbindrule 464

sp_volchanged 466
sp_who 469

CHAPTER 2 **Catalog Stored Procedures 473**
Overview 473
Specifying optional parameters 474
Pattern matching 475
System procedure tables 475
ODBC datatypes 476
sp_column_privileges 477
sp_columns 479
sp_databases 481
sp_datatype_info 482
sp_fkeys 483
sp_pkeys 485
sp_server_info 486
sp_special_columns 489
sp_sproc_columns 491
sp_statistics 493
sp_stored_procedures 495
sp_table_privileges 496
sp_tables 497

CHAPTER 3 **System Extended Stored Procedures 499**
Overview 499
Permissions on system ESPs 500
DLLs associated with system ESPs 500
Using system ESPs 500
xp_cmdshell 501
xp_deletemail 504
xp_enumgroups 505
xp_findnextmsg 506
xp_logevent 507
xp_readmail 508
xp_sendmail 511
xp_startmail 515
xp_stopmail 516

CHAPTER 4 **dbcc Stored Procedures 517**
Overview 517
Specifying the object name and date 518
    Specifying the object name 518

Specifying the date 518
sp_dbcc_alterws 520
sp_dbcc_configreport 521
sp_dbcc_createws 522
sp_dbcc_deletedb 524
sp_dbcc_deletehistory 525
sp_dbcc_differentialreport 527
sp_dbcc_evaluatedb 529
sp_dbcc_faultreport 531
sp_dbcc_fullreport 534
sp_dbcc_recommendations 535
sp_dbcc_runcheck 536
sp_dbcc_statisticsreport 537
sp_dbcc_summaryreport 540
sp_dbcc_updateconfig 543

Index 547

About This Book

The *Adaptive Server Reference Manual* includes four guides to Sybase® Adaptive Server® Enterprise and the Transact-SQL® language:

- *Building Blocks* describes the “parts” of Transact-SQL: datatypes, built-in functions, global variables, expressions and identifiers, reserved words, and SQLSTATE errors. Before you can use Transact-SQL sucessfully, you need to understand what these building blocks do and how they affect the results of Transact-SQL statements.
- *Commands* provides reference information about the Transact-SQL commands, which you use to create statements.
- *Procedures* provides reference information about system procedures, catalog stored procedures, extended stored procedures, and dbcc stored procedures. All procedures are created using Transact-SQL statements.
- *Tables* provides reference information about the system tables, which store information about your server, databases, users, and other details of your server. It also provides information about the tables in the dbccdb and dbccalt databases.

Audience

The *Adaptive Server Reference Manual* is intended as a reference tool for Transact-SQL users of all levels.

How to use this book

- Chapter 1, “System Procedures” lists the Adaptive Server system procedures in a table that provides the name and a brief description. Click on a procedure name in the table to go directly to the procedure.
- Chapter 2, “Catalog Stored Procedures” contains reference pages for Adaptive Server catalog stored procedures.
- Chapter 3, “System Extended Stored Procedures” contains reference pages for Adaptive Server system extended stored procedures.
- Chapter 4, “dbcc Stored Procedures” contains reference pages for Adaptive Server dbcc stored procedures.

Related documents

The Sybase Adaptive Server Enterprise documentation set consists of the following:

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

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library.

- The *Installation Guide* for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 12.5.1, the system changes added to support those features, and the changes that may affect your existing applications.
- *ASE Replicator User's Guide* – describes how to use the ASE Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.
- *Component Integration Services User's Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Configuring Adaptive Server Enterprise* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.
- *EJB Server User's Guide* – explains how to use EJB Server to deploy and execute Enterprise JavaBeans in Adaptive Server.
- *Error Messages and Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server User's Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as data types, functions, and stored procedures in the Adaptive Server database.

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

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

Other sources of information

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

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).

- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

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

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

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ Finding the latest information on product certifications

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

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

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

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

Sybase EBFs and software maintenance

❖ Finding the latest information on EBFs and software maintenance

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2. Select EBFs/Maintenance. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).
3. Select a product.
4. Specify a time frame and click Go.
5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following sections describe conventions used in this manual.

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

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

Table 1: Font and syntax conventions for this manual

Element	Example
Command names, command options, utility names, utility options, and other keywords are in "command" font (Arial, 8 point).	select sp_configure
Database names, datatypes, file names and path names are in "database object" font (Arial, 8 point).	master database
Book names, file names, variables, and path names are in italics.	*System Administration Guide* *sql.ini* file *column_name* *$SYBASE/ASE* directory
Variables, or words that stand for values that you fill in, are in "variable" font (Italics).	select *column_name* from *table_name* where *search_conditions*
Type parentheses as part of the command.	compute *row_aggregate* (*column_name*)
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates "is defined as".	::=
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	{cash, check, credit}

Element	Example
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	[cash | check | credit]
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	cash, check, credit
The pipe or vertical bar (|) means you may select only one of the options shown.	cash | check | credit
An ellipsis (...) means that you can repeat the last unit as many times as you like.	buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]... You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

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

  sp_dropdevice [*device_name*]

  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. Italic font shows user-supplied words.

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

```
select * from publishers
```

- Examples of output from the computer appear as follows:

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

(3 rows affected)
```

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

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

If you need help

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

CHAPTER 1

System Procedures

This chapter describes the system procedures, which are Sybase-supplied stored procedures used for updating and getting reports from system tables. "List of system procedures" on page 5 lists the system procedures described in this volume.

Topics covered are:

Topics	Page
Introduction to system procedures	1
Permissions on system procedures	2
Executing system procedures	2
Entering parameter values	3
Messages	4
System procedure tables	5
List of system procedures	5

Introduction to system procedures

System procedures are created by installmaster at installation. They are located in the sybsystemprocs database, and owned by the System Administrator.

Some system procedures can be run only in a specific database, but many of them can be run in any database. You can create your own system procedures that can be executed from any database. For more information, see the *System Administration Guide*.

All system procedures execute at isolation level 1.

All system procedures report a return status. The following example means that the procedure executed successfully:

```
    return status = 0
```

The examples in this book do not include the return status.

Beginning with Adaptive Server version 12.5.1, you can declare up to 10,000 variables in a stored procedure. In earlier versions of Adaptive Server, the limit was 2,000.

Permissions on system procedures

Permissions for system procedures are set in the sybsystemprocs database.

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, provided that the permission was granted in sybsystemprocs. A user must have permission to execute a system procedure either in all databases or in none of them.

A user who is not listed in sybsystemprocs..sysusers is treated as a "guest" user in sybsystemprocs and is 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.

Executing system procedures

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

To run a system procedure in a specific database, either:

- Open that database with the use command and execute the procedure, or
- Qualify the procedure name with the database name.

For example, the user-defined system procedure *sp_foo*, which executes the db_name system function, returns the name of the database in which it is executed. When executed in the pubs2 database, it returns the value "pubs2":

```
exec pubs2..sp_foo
------------------------------
pubs2
(1 row affected, return status = 0)
```

When executed in sybsystemprocs, it returns the value "sybsystemprocs":

```
exec sybsystemprocs..sp_foo
------------------------------
sybsystemprocs
(1 row affected, return status = 0)
```

Entering parameter values

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.

If a procedure has multiple optional parameters, you can supply parameters in the following form instead of supplying all the parameters:

```
@parametername = value
```

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 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.

SQL has no rules about the number of words you can put on a line or where you must break a line. If you issue a system procedure followed by a command, Adaptive Server attempts to execute the system procedure, then the command. For example, if you execute the following command, Adaptive Server returns the output from sp_help, then runs the checkpoint command:

```
    sp_help checkpoint
```

If you specify more parameters than the number of parameters expected by the system procedure, the extra parameters are ignored by Adaptive Server.

Messages

System procedures return informational and error messages, which are listed with each procedure in this book. System procedure error messages start at error number 17000.

Error messages from the functions and commands included in a procedure are documented in *Troubleshooting and Error Messages Guide*.

System procedure tables

Several **system procedure tables** in the master database, such as spt_values, spt_committab, spt_monitor, and spt_limit_types, are used by system procedures to convert internal system values (for example, status bits) into human-readable format.

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.

In addition, some system procedures create and then drop temporary tables.

List of system procedures

Table 1-1 provides a brief description of each system procedure.

Table 1-1: System procedures

Procedure	Description
sp_activeroles on page 15	Displays all active roles granted to a user's login.
sp_addalias on page 16	Allows an Adaptive Server user to be known in a database as another user.
sp_addauditrecord on page 17	Allows users to enter user-defined audit records (comments) into the audit trail.
sp_addaudittable on page 19	Adds another system audit table after auditing is installed.
sp_addengine on page 21	Adds an engine to an existing engine group or, if the group does not exist, creates an engine group and adds the engine.
sp_addexeclass on page 22	Creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures.
sp_addextendedproc on page 23	Creates an extended stored procedure (ESP) in the master database.
sp_addexternlogin on page 25	Creates an alternate login account and password to use when communicating with a remote server through Component Integration Services.
sp_addgroup on page 28	Adds a group to a database. Groups are used as collective names in granting and revoking privileges.
sp_addlanguage on page 29	Defines the names of the months and days, and the date format, for an alternate language.
sp_addlogin on page 32	Adds a new user account to Adaptive Server.
sp_addmessage on page 35	Adds user-defined messages to sysusermessages for use by stored procedure print and raiserror calls and by sp_bindmsg.
sp_addobjectdef on page 37	Specifies the mapping between a local table and an external storage location.
sp_add_qpgroup on page 40	Adds an abstract plan group.

Procedure	**Description**
sp_addremotelogin on page 41	Authorizes a new remote server user by adding an entry to master.dbo.sysremotelogins.
sp_add_resource_limit on page 44	Creates a limit on the amount of server resources that a login or application can use to execute a query, query batch, or transaction.
sp_addsegment on page 49	Defines a segment on a database device in the current database.
sp_addserver on page 51	Defines a remote server or defines the name of the local server.
sp_addthreshold on page 54	Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, Adaptive Server executes the associated stored procedure.
sp_add_time_range on page 58	Adds a named time range to Adaptive Server.
sp_addtype on page 61	Creates a user-defined datatype.
sp_addumpdevice on page 65	Adds a dump device to Adaptive Server.
sp_adduser on page 67	Adds a new user to the current database.
sp_altermessage on page 69	Enables and disables the logging of a specific system-defined or user-defined message in the Adaptive Server error log.
sp_audit on page 70	Allows a System Security Officer to configure auditing options.
sp_autoconnect on page 76	Defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.
sp_bindcache on page 78	Binds a database, table, index, text object, or image object to a data cache.
sp_bindefault on page 82	Binds a user-defined default to a column or user-defined datatype.
sp_bindexeclass on page 84	Associates an execution class with a client application, login, or stored procedure.
sp_bindmsg on page 87	Binds a user message to a referential integrity constraint or check constraint.
sp_bindrule on page 88	Binds a rule to a column or user-defined datatype.
sp_cacheconfig on page 90	Creates, configures, reconfigures, drops, and provides information about data caches.
sp_cachestrategy on page 98	Enables or disables prefetching (large I/O) and MRU cache replacement strategy for a table, index, text object, or image object.
sp_changedbowner on page 101	Changes the owner of a database.
sp_changegroup on page 102	Changes a user's group.
sp_checknames on page 103	Checks the current database for names that contain characters not in the 7-bit ASCII set.
sp_checkreswords on page 104	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_checksource on page 117	Checks for the existence of the **source text** of the **compiled object**.

Procedure	Description
sp_chgattribute on page 119	Changes the max_rows_per_page value for future space allocations of a table or index.
sp_clearpsexe on page 123	Clears the execution attributes of the client application, login, or stored procedure that was set by sp_setpsexe.
sp_clearstats on page 124	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_client_addr on page 126	Displays the IP address of every Adaptive Server task with an attached client application, including the spid and the client host name.
sp_cmp_all_qplans on page 128	Compares all abstract plans in two abstract plan groups.
sp_cmp_qplans on page 130	Compares two abstract plans.
sp_commonkey on page 132	Defines a common key—columns that are frequently joined—between two tables or views.
sp_companion on page 134	Performs cluster operations such as configuring Adaptive Server as a secondary companion in a high availability system and moving a companion server from one failover mode to another
sp_configure on page 137	Displays or changes configuration parameters.
sp_copy_all_qplans on page 143	Copies all plans for one abstract plan group to another group.
sp_copy_qplan on page 144	Copies one abstract plan to an abstract plan group.
sp_countmetadata on page 145	Displays the number of indexes, objects, or databases in Adaptive Server.
sp_cursorinfo on page 147	Reports information about a specific cursor or all cursors that are active for your session.
sp_dbextend on page 150	Allows you to: • Install automatic database expansion procedures on database/segment pairs and devices. • Define site-specific policies for individual segments and devices. • Simulate execution of the database expansion machinery, to study the operation before engaging large volume loads.
sp_dboption on page 157	Displays or changes database options.
sp_dbrecovery_order on page 165	Specifies the order in which user databases are recovered and lists the user-defined recovery order of a database or all databases.
sp_dbremap on page 167	Forces Adaptive Server to recognize changes made by alter database. Run this procedure only when instructed to do so by an Adaptive Server message.
sp_defaultloc on page 168	**Component Integration Services only** Defines a default storage location for objects in a local database.
sp_depends on page 171	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.
sp_deviceattr on page 177	Changes the dsync setting of an existing database device file.

Procedure	Description
sp_diskdefault on page 179	Specifies 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_displayaudit on page 181	Displays the status of audit options.
sp_displaylevel on page 185	Sets or shows which Adaptive Server configuration parameters appear in sp_configure output.
sp_displaylogin on page 186	Displays information about a login account.
sp_displayroles on page 189	Displays all roles granted to another role, or displays the entire hierarchy tree of roles in table format.
sp_dropalias on page 191	Removes the alias user name identity established with sp_addalias.
sp_drop_all_qplans on page 192	Deletes all abstract plans in an abstract plan group.
sp_dropdevice on page 193	Drops an Adaptive Server database device or dump device.
sp_dropengine on page 194	Drops an engine from a specified engine group or, if the engine is the last one in the group, drops the engine group.
sp_dropexeclass on page 195	Drops a user-defined execution class.
sp_dropextendedproc on page 196	Removes an ESP from the master database.
sp_dropexternlogin on page 197	**Component Integration Services only** Drops the definition of a remote login previously defined by sp_addexternlogin.
sp_dropglockpromote on page 198	Removes lock promotion values from a table or database.
sp_dropgroup on page 199	Drops a group from a database.
sp_dropkey on page 200	Removes a key defined with sp_primarykey, sp_foreignkey, or sp_commonkey from the syskeys table.
sp_droplanguage on page 202	Drops an alternate language from the server and removes its row from master.dbo.syslanguages.
sp_droplogin on page 203	Drops an Adaptive Server user login by deleting the user's entry in master.dbo.syslogins.
sp_dropmessage on page 204	Drops user-defined messages from sysusermessages.
sp_dropobjectdef on page 205	**Component Integration Services only** Deletes the external storage mapping provided for a local object
sp_drop_qpgroup on page 207	Drops an abstract plan group.
sp_drop_qplan on page 208	Drops an abstract plan.
sp_dropremotelogin on page 209	Drops a remote user login.
sp_drop_resource_limit on page 210	Removes one or more resource limits from Adaptive Server.
sp_droprowlockpromote on page 213	Removes row lock promotion threshold values from a database or table.
sp_dropsegment on page 214	Drops a segment from a database or unmaps a segment from a particular database device.
sp_dropserver on page 216	Drops a server from the list of known servers.

Procedure	Description
sp_dropthreshold on page 217	Removes a free-space threshold from a segment.
sp_drop_time_range on page 218	Removes a user-defined time range from Adaptive Server.
sp_droptype on page 219	Drops a user-defined datatype.
sp_dropuser on page 220	Drops a user from the current database.
sp_dumpoptimize on page 221	Specifies the amount of data dumped by Backup Server during the dump database operation.
sp_engine on page 226	Enables you to bring an engine online or offline.
sp_estspace on page 229	Estimates the amount of space required for a table and its indexes, and the time needed to create the index.
sp_export_qpgroup on page 234	Exports all plans for a specified user and abstract plan group to a user table.
sp_extendsegment on page 235	Extends the range of a segment to another database device.
sp_extengine on page 236	Starts and stops EJB Server. Displays status information about EJB Server.
sp_familylock on page 237	Reports information about all the locks held by a family (coordinating process and its worker processes) executing a statement in parallel.
sp_find_qplan on page 240	Finds an abstract plan, given a pattern from the query text or plan text.
sp_fixindex on page 242	Repairs the index on one of your system tables when it has been corrupted.
sp_flushstats on page 244	Flushes statistics from in-memory storage to the systabstats system table.
sp_forceonline_db on page 245	Provides access to all the pages in a database that were previously taken offline by recovery.
sp_forceonline_object	Provides access to an index previously marked suspect by recovery.
sp_forceonline_page on page 248	Provides access to pages previously taken offline by recovery.
sp_foreignkey on page 250	Defines a foreign key on a table or view in the current database.
sp_freedll on page 252	Unloads a dynamic link library (DLL) that was previously loaded into XP Server memory to support the execution of an ESP.
sp_getmessage on page 253	Retrieves stored message strings from sysmessages and sysusermessages for print and raiserror statements.
sp_grantlogin on page 254	**Windows NT only** When Integrated Security mode or Mixed mode (with Named Pipes) is active, assigns Adaptive Server roles or default permissions to Windows NT users and groups.
sp_ha_admin on page 256	Performs administrative tasks on Adaptive Servers configured with Sybase Failover in a high availability system. sp_ha_admin is installed with the *installhavss* script (*insthasv* on Windows NT).
sp_help on page 257	Reports information about a database object (any object listed in sysobjects) and about Adaptive Server-supplied or user-defined datatypes.
sp_helpartition on page 264	Lists the first page and the control page for each partition in a partitioned table.

Procedure	Description
sp_helpcache on page 267	Displays information about the objects that are bound to a data cache or the amount of overhead required for a specified cache size.
sp_helpconfig on page 269	Reports help information on configuration parameters.
sp_helpconstraint on page 274	Reports information about integrity constraints used in the specified tables.
sp_helpdb on page 278	Reports information about a particular database or about all databases.
sp_helpdevice on page 281	Reports information about a particular device or about all Adaptive Server database devices and dump devices.
sp_helpextendedproc on page 283	Displays ESPs registered in the current database, along with their associated DLL files.
sp_helpexternlogin on page 284	**Component Integration Services only** Reports information about external login names.
sp_helpgroup on page 285	Reports information about a particular group or about all groups in the current database.
sp_helpindex on page 286	Reports information about the indexes created on a table.
sp_helpjava on page 288	Displays information about Java classes and associated JARs that are installed in the database.
sp_helpjoins on page 290	Lists the columns in two tables or views that are likely join candidates.
sp_helpkey on page 292	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 on page 294	Reports information about a particular alternate language or about all languages.
sp_helplog on page 295	Reports the name of the device that contains the first page of the transaction log.
sp_helpobjectdef on page 296	**Component Integration Services only** Reports information about remote object definitions. Shows owners, objects, type, and definition.
sp_help_qpgroup on page 297	Reports information on an abstract plan group.
sp_help_qplan on page 299	Reports information about an abstract plan.
sp_helpremotelogin on page 300	Reports information about a particular remote server's logins or about all remote servers' logins.
sp_help_resource_limit on page 301	Reports information about all resource limits, limits for a given login or application, limits in effect at a given time or day of the week, or limits with a given scope or action.
sp_helprotect on page 304	Reports information about permissions for database objects, users, groups, or roles.
sp_helpsegment on page 308	Reports information about a particular segment or about all segments in the current database.
sp_helpserver on page 311	Reports information about a particular remote server or about all remote servers.
sp_helpsort on page 312	Displays Adaptive Server's default sort order and character set.

Procedure	Description
sp_helptext on page 314	Prints the text of a system procedure, trigger, view, default, rule, or integrity check constraint, and adds the number parameter, which is an integer identifying an individual procedure, when *objname* represents a group of procedures. This parameter tells sp_helptext to display the source text for a specified procedure in the group.
sp_helpthreshold on page 316	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 on page 317	Reports information about a particular user or about all users in the current database.
sp_hidetext on page 318	Hides the **source text** for the specified **compiled object**.
sp_import_qpgroup on page 320	Imports abstract plans from a user table into an abstract plan group.
sp_indsuspect on page 322	Checks user tables for indexes marked as suspect during recovery following a sort order change.
sp_bindcache on page 78	Creates or lists an LDAP URL search string; verifies an LDAP URL search string or login.
sp_listsuspect_db on page 329	Lists all databases that have offline pages because of corruption detected on recovery.
sp_listsuspect_object on page 330	Lists all indexes in a database that are currently offline because of corruption detected on recovery.
sp_listsuspect_page on page 331	Lists all pages that are currently offline because of corruption detected on recovery.
sp_lock on page 332	Reports information about processes that currently hold locks.
sp_locklogin on page 336	Locks an Adaptive Server account so that the user cannot log in, or displays a list of all locked accounts.
sp_logdevice on page 338	Moves the transaction log of a database with log and data on the same device to a separate database device.
sp_loginconfig on page 340	**Windows NT only** Displays the value of one or all integrated security parameters.
sp_logininfo on page 342	**Windows NT only** Displays all roles granted to Windows NT users and groups with sp_grantlogin.
sp_logiosize on page 343	Changes the log I/O size used by Adaptive Server to a different memory pool when it is doing I/O for the transaction log of the current database.
sp_modifylogin on page 346	Modifies the default database, default language, default role activation, or full name for an Adaptive Server login account.
sp_modify_resource_limit on page 349	Changes a resource limit by specifying a new limit value or the action to take when the limit is exceeded, or both.
sp_modify_time_range on page 352	Changes the start day, start time, end day, and/or end time associated with a named time range.
sp_modifystats on page 354	Allows the System Administrator to modify the density values of a column—or columns—in sysstatistics.

Procedure	Description
sp_modifythreshold on page 357	Modifies a threshold by associating it with a different threshold procedure, free-space level, or segment name. You *cannot* use sp_modifythreshold to change the amount of free space or the segment name for the last-chance threshold.
sp_monitor on page 361	Displays statistics about Adaptive Server.
sp_monitorconfig on page 364	Monitors more than 30 resources compared to the 6 resources it monitored in earlier versions.
sp_object_stats on page 371	Shows lock contention, lock wait-time, and deadlock statistics for tables and indexes.
sp_passthru on page 374	**Component Integration Services only** Allows the user to pass a SQL command buffer to a remote server.
sp_password on page 376	Adds or changes a password for an Adaptive Server login account.
sp_placeobject on page 378	Puts future space allocations for a table or an index on a particular segment.
sp_plan_dbccdb on page 380	Recommends suitable sizes for new dbccdb and dbccalt databases, lists suitable devices for dbccdb and dbccalt, and suggests a cache size and a suitable number of worker processes for the target database.
sp_poolconfig on page 382	Creates, drops, resizes, and provides information about memory pools within data caches.
sp_primarykey on page 387	Defines a primary key on a table or view.
sp_processmail on page 388	**Windows NT only** Reads, processes, sends, and deletes messages in the Adaptive Server message inbox.
sp_procxmode on page 390	Displays or changes the transaction modes associated with stored procedures.
sp_recompile on page 392	Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.
sp_remap on page 393	Remaps a stored procedure, trigger, rule, default, or view from releases later than 4.8 and earlier than 10.0 to be compatible with releases 10.0 and later. Use sp_remap on pre-release 11.0 objects that the release 11.0 upgrade procedure failed to remap.
sp_remoteoption on page 394	Displays or changes remote login options.
sp_remotesql on page 396	**Component Integration Services only** Establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client.
sp_rename on page 398	Changes the name of a user-created object or user-defined datatype in the current database.
sp_renamedb on page 400	Changes the name of a database. You *cannot* rename system databases or databases with external referential integrity constraints.
sp_rename_qpgroup on page 402	Renames an abstract plan group.
sp_reportstats on page 403	Reports statistics on system usage.

Procedure	Description
sp_revokelogin on page 405	**Windows NT only** When Integrated Security mode or Mixed mode (with Named Pipes) is active, revokes Adaptive Server roles and default permissions from Windows NT users and groups.
sp_role on page 406	Grants or revokes system roles to an Adaptive Server login account.
sp_sendmsg on page 407	Sends a message to a User Datagram Protocol (UDP) port.
sp_serveroption on page 409	Displays or changes remote server options.
sp_setlangalias on page 413	Assigns or changes the alias for an alternate language.
sp_setpglockpromote on page 414	Sets or changes the lock promotion thresholds for a database, for a table, or for Adaptive Server.
sp_setpsexe on page 416	Sets custom execution attributes "on the fly" for an active client application, login, or stored procedure.
sp_set_qplan on page 417	Changes the text of the abstract plan of an existing plan without changing the associated query.
sp_setrowlockpromote on page 418	Sets or changes row-lock promotion thresholds for a datarows-locked table, for all datarows-locked tables in a database, or for all datarows-locked tables on a server.
sp_setsuspect_granularity on page 420	Displays and sets the recovery fault isolation mode.
sp_setsuspect_threshold on page 423	On recovery, sets the maximum number of suspect pages that Adaptive Server will allow in the specified database before taking the entire database offline.
sp_showcontrolinfo on page 424	Displays information about engine group assignments, bound client applications, logins, and stored procedures.
sp_showexeclass on page 426	Displays the execution class attributes and the engines in any engine group associated with the specified execution class.
sp_showplan on page 427	Displays the query plan for any user connection for the current SQL statement (or a previous statement in the same batch). The query plan is displayed in showplan format.
sp_showpsexe on page 429	Displays execution class, current priority, and affinity for all processes running on Adaptive Server.
sp_spaceused on page 430	Displays estimates of 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_ssladmin on page 433	Adds, deletes, or displays a list of server certificates for Adaptive Server.
sp_syntax on page 435	Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines, depending on which products and corresponding sp_syntax scripts exist on Adaptive Server.
sp_sysmon on page 437	Displays performance information.

Procedure	**Description**
sp_tempdb	Creates the default temporary database group, binds temporary databases to the default temporary database group, binds users and applications to the default temporary database group or to specific temporary databases, and provides the binding interface for maintaining bindings in sysattributes that are related to the multiple temporary database.
sp_thresholdaction on page 448	Executes automatically when the number of free pages on the log segment falls below the last-chance threshold, unless the threshold is associated with a different procedure. Sybase does not provide this procedure.
sp_transactions on page 450	Reports information about active transactions.
sp_unbindcache on page 457	Unbinds a database, table, index, text object, or image object from a data cache.
sp_unbindcache_all on page 459	Unbinds all objects that are bound to a cache.
sp_unbindefault on page 460	Unbinds a created default value from a column or from a user-defined datatype.
sp_unbindexeclass on page 461	Removes the execution class attribute previously associated with an client application, login, or stored procedure for the specified scope.
sp_unbindmsg on page 463	Unbinds a user-defined message from a constraint.
sp_unbindrule on page 464	Unbinds a rule from a column or from a user-defined datatype.
sp_volchanged on page 466	Notifies the Backup Server™ that the operator performed the requested volume handling during a dump or load.
sp_who on page 469	Reports information about all current Adaptive Server users and processes or about a particular user or process.

sp_activeroles

Description
Displays all active roles.

Syntax
sp_activeroles [expand_down]

Parameters
expand_down
shows the hierarchy tree of all active roles contained by your roles.

Examples
Example 1

```
sp_activeroles
```

```
Role Name
------------------------------
sa_role
sso_role
oper_role
replication_role
```

Example 2

```
sp_activeroles expand_down
```

```
Role Name       Parent Role Name              Level
------------------------------- ------------------
sa_role         NULL                              1
doctor_role     NULL                              1
oper_role       NULL                              1
```

Usage
- sp_activeroles displays all your active roles and all roles contained by those roles.

Permissions
Any user can execute sp_activeroles.

See also
Documents For information about creating, managing, and using roles, see the *System Administration Guide*.

Commands alter role, create role, drop role, grant, revoke, set

Functions mut_excl_roles, proc_role, role_contain, role_name

System procedures sp_displayroles

sp_addalias

Description

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

Syntax

sp_addalias *loginame*, *name_in_db*

Parameters

loginame

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 *loginame* to. The name must exist in both master.dbo.syslogins and in the sysusers table of the current database.

Examples

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”:

```
sp_addalias victoria, albert
```

Usage

- Executing sp_addalias maps one user to another in the current database. The mapping is shown in sysalternates, where the two users’ suids (system user IDs) are connected.
- A user can 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, Adaptive Server checks sysusers to confirm that the user is listed there. If the user is not listed there, Adaptive Server then checks sysalternates. If the user’s suid is listed in sysalternates, mapped to a database user’s suid, Adaptive Server treats the first user as the second user while using the database.

  If the user named in *loginame* is in the database’s sysusers table, Adaptive Server does not use the user’s alias identity, because it checks sysusers and finds the loginame before checking sysalternates, where the alias is listed.

Permissions

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

See also

Command use

System procedures sp_addlogin, sp_adduser, sp_dropalias, sp_helpuser

sp_addauditrecord

Description

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

Syntax

sp_addauditrecord [*text* [, *db_name* [, *obj_name*
    [, *owner_name* [, *dbid* [, *objid*]]]]]]

Parameters

text
is the text of the message to add to the current audit table. The text is inserted into the extrainfo field of the table.

db_name
is the name of the database referred to in the record. The name is inserted into the dbname field of the current audit table.

obj_name
is the name of the object referred to in the record. The name is inserted into the objname field of the current audit table.

owner_name
is the owner of the object referred to in the record. The name is inserted into the objowner field of the current audit table.

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 the current audit table.

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 the current audit table.

Examples

Example 1 Adds "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." to the extrainfo field; "corporate" to the dbname field; "payroll" to the objname field; "dbo" to the objowner field; "10" to the dbid field, and "1004738270" to the objid field of the current audit table:

```
sp_addauditrecord "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.", "corporate", "payroll", "dbo", 10,
1004738270
```

Example 2 Adds this record to the audit trail. This example uses parameter names with the @ prefix, which allows you to leave some fields empty:

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

Usage

- Adaptive Server writes all audit records to the current audit table. The current audit table is determined by the value of the current audit table configuration parameter, set with sp_configure. An installation can have up to eight system audit tables, named sysaudits_01, sysaudits_02, and so forth, through sysaudits_08.

  Note The records actually are first stored in the in-memory audit queue, and the audit process later writes the records from the audit queue to the current audit table. Therefore, you cannot count on an audit record being stored immediately in the audit table.

- You can use sp_addauditrecord if:
  - You have been granted execute permission on sp_addauditrecord – no special role is required
  - Auditing is enabled – a System Security Officer used sp_configure to turn on the auditing configuration parameter
  - The adhoc option of sp_audit is set to on

Permissions

Only a System Security Officer can execute sp_addauditrecord. The Database Owner of sybsecurity (who must also be a System Security Officer) can grant execute permission to other users.

See also

System procedure sp_audit

sp_addaudittable

Description

Adds another system audit table after auditing is installed.

Syntax

sp_addaudittable *devname*

Parameters

devname
is the name of the device for the audit table. Specify a device name or specify “default”. If you specify “default”, Adaptive Server creates the audit table on the same device as the sybsecurity database. Otherwise, Adaptive Server creates the table on the device you specify.

Examples

Example 1 Creates a system audit table on auditdev2. If only one system audit table (sysaudits_01) exists when you execute the procedure, Adaptive Server names the new audit table sysaudits_02 and places it on its own segment, called aud_seg_02, on auditdev2:

```
sp_addaudittable auditdev2
```

Example 2 Creates a system audit table on the same device as the sybsecurity database. If two system audit tables (sysaudits_01 and sysaudits_02) exist when you execute the procedure, Adaptive Server names the new audit table sysaudits_03 and places it on its own segment, called aud_seg_03, on the same device as the sybsecurity database:

```
sp_addaudittable "default"
```

Usage

- Auditing must already be installed when you run sp_addaudittable. Follow this procedure to add a system audit table:

  a. Create the device for the audit table, using disk init. For example, run a command like this for UNIX:

  ```
  disk init name = "auditdev2",
  physname = "/dev/rxyla",
  size = “5K”
  ```

  b. Add the device to the sybsecurity database with the alter database command. For example, to add auditdev2 to the sybsecurity database, use:

  ```
  alter database sybsecurity on auditdev2
  ```

  c. Execute sp_addaudittable to create the table.

- Adaptive Server names the new system audit table and the new segment according to how many audit tables are already defined. For example, if five audit tables are defined before you execute the procedure, Adaptive Server names the new audit table sysaudits_06 and the new segment aud_seg_06. If you specify “default”, Adaptive Server places the segment on the same device as the sybsecurity database. Otherwise, Adaptive Server places the segment on the device you name.
- A maximum of eight audit tables is allowed. If you already have eight audit tables, and you attempt to execute sp_addaudittable to add another one, Adaptive Server displays an error message.
- For information about how to install auditing, see the installation documentation for your platform. See the *System Administration Guide* for information on how to use auditing.

Permissions

Only a uses who is both a System Administrator and a System Security Officer to execute sp_addaudittable.

See also

System procedure sp_audit

sp_addengine

Description

Adds an engine to an existing engine group or, if the group does not exist, creates an engine group and adds the engine.

Syntax

sp_addengine *engine_number*, *engine_group*

Parameters

engine_number
is the number of the engine you are adding to the group. Legal values are between 0 and a maximum equal to the number of configured online engines minus one.

engine_group
is the name of the engine group to which you are adding the engine. If engine_group does not exist, Adaptive Server creates it and adds the engine to it. Engine group names must conform to the rules for identifiers. For details, see Chapter 4, "Expressions, Identifiers, and Wildcard Characters" in *Reference Manual: Building Blocks*.

Examples

If no engine group is called DS_GROUP, this statement establishes the group. If DS_GROUP already exists, this statement adds engine number 2 to that group:

```
sp_addengine 2, DS_GROUP
```

Usage

- sp_addengine creates a new engine group if the value of engine_group does not already exist.
- The engine groups ANYENGINE and LASTONLINE are predefined. ANYENGINE includes all existing engines. LASTONLINE specifies the engine with highest engine number. A System Administrator can create additional engine groups. You cannot modify predefined engine groups.
- As soon as you use sp_bindexeclass to bind applications or logins to an execution class associated with engine_group, the associated process may start running on engine_number.
- Prior to making engine affinity assignments, study the environment and consider the number of non-preferred applications and the number of Adaptive Server engines available. See the *Performance and Tuning Guide* for more information about non-preferred applications.

Permissions

Only a System Administrator can execute sp_addengine.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass

sp_addexeclass

Description

Creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures.

Syntax

sp_addexeclass *classname*, *priority*, *timeslice*, *engine_group*

Parameters

classname
is the name of the new execution class.

priority
is the priority value with which to run the client application, login, or stored procedure after it is associated with this execution class. Legal values are HIGH, LOW, and MEDIUM.

timeslice
is the time unit assigned to processes associated with this class. Adaptive Server currently ignores this parameter.

engine_group
identifies an existing group of engines on which processes associated with this class can run.

Examples

Defines a new execution class called DS with a *priority* value of LOW and associates it with the engine group DS_GROUP:

```
sp_addexeclass "DS", "LOW", 0, "DS_GROUP"
```

Usage

- sp_addexeclass creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures. If the class already exists, the class attribute values are updated with the values supplied by the user.
- Use the predefined engine group parameter ANYENGINE if you do not want to restrict the execution object to an engine group.
- Use sp_addengine to define engine groups. Use sp_showexeclass to display execution class attributes and the engines in any engine group associated with the specified execution class. sp_showcontrolinfo lists the existing engine groups.

Permissions

Only a System Administrator can execute sp_addexeclass.

See also

System procedures sp_addengine, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass

sp_addextendedproc

Description

Creates an extended stored procedure (ESP) in the master database.

Syntax

sp_addextendedproc *esp_name*, *dll_name*

Parameters

esp_name
is the name of the extended stored procedure. This name must be identical to the name of the procedural language function that implements the ESP. *esp_name* must be a valid Adaptive Server identifier.

dll_name
is the name of the dynamic link library (DLL) file containing the function specified by *esp_name*. The *dll_name* can be specified with no extension or with its platform-specific extension, such as *.dll* on Windows NT or *.so* on Sun Solaris. If an extension is specified, the *dll_name* must be enclosed in quotation marks.

Examples

Registers an ESP for the function named my_esp, which is in the *sqlsrvdll.dll* file. The name of the resulting ESP database object is also xp_echo:

```
sp_addextendedproc my_esp, "sqlsrvdll.dll"
```

Usage

- Execute sp_addextendedproc from the master database.
- You can only use sp_addextendedproc to add extended stored procedures that take no parameters. If your extended stored procedure requires a formal parameter list, you must use the create procedure command with the as external name option, together with the complete parameter list.
- The *esp_name* is case sensitive. It must match the name of the function in the DLL.
- The DLL represented by *dll_name* must reside on the server machine on which the ESP is being created and the DLL directory must be in:
  - Windows NT – *$PATH*
  - Compaq Tru64 – *$LD_LIBRARY_PATH*
  - HP – *$SH_LIBRARY_PATH*

  If the file is not found, the search mechanism also searches *$SYBASE/dll* on Windows NT and *$SYBASE/lib* on other platforms.
- *On Windows NT* – an ESP function should not call a C run-time signal routine. This can cause XP Server to fail, because Open Server™ does not support signal handling on Windows NT.

Permissions

Only a System Administrator can execute sp_addextendedproc.

See also

Commands create procedure

System procedures sp_dropextendedproc, sp_helpextendedproc

sp_addexternlogin

Description

Component Integration Services only Creates an alternate login account and password to use when communicating with a remote server through Component Integration Services.

Syntax

```
sp_addexternlogin remote_server, login_name, remote_name
      [, remote_password] [role_name]
```

Parameters

remote_server
is the name of the remote server. The *remote_server* must be known to the local server by an entry in the master.dbo.sysservers table.

login_name
is an account known to the local server. *login_name* must be represented by an entry in the master.dbo.syslogins table. The “sa” account, the “sso” account, and the *login_name* account are the only users authorized to modify remote access for a given local user.

remote_name
is an account known to the *remote_server* and must be a valid account on the node where the *remote_server* runs. This is the account used for logging into the *remote_server*.

remote_password
is the password for *remote_name*.

role_name
is the Adaptive Server user’s assigned role. If *role_name* is specified, *login_name* is ignored.

Examples

Example 1 Tells the local server that when the login name “bobj” logs in, access to the remote server OMNI1012 is by the remote name “jordan” and the remote password “hitchpost”. Only the “bobj” account, the “sa” account, and the “sso” account have the authority to add or modify a remote login for the login name “bobj”:

```
sp_addexternlogin OMNI1012, bobj, jordan, hitchpost
```

Example 2 Shows a many-to-one mapping so that all Adaptive Server Enterprise users that need a connection to DB2 can be assigned the same name and password:

```
sp_addexternlogin DB2, NULL, login2, password2
```

Example 3 Adaptive Server Enterprise roles can also be assigned remote logins. With this capability, anyone with a particular role can be assigned a corresponding login name and password for a given remote server:

```
sp_addexternlogin DB2, NULL, login3, password3, role
```

Usage

- sp_addexternlogin assigns an alternate login name and password to be used when communicating with a remote server. It stores the password internally in encrypted form.

Note You can use sp_addexternlogin only when Component Integration Services is installed and configured.

- Mappings can be one-to-one (for specific users), role-to-one (role-based) , many-to-one (server-based), or based on the client login and password from the TDS loginrec.
- The login and password have a many to one mapping. That is, you can assign all the users who need to log into a remote server the same name and password.
- When several external logins are set for a user, the following precedence will be followed for user connections to a remote server. 1) one-to-one mapping, 2) if there is no one-to-one mapping, active role is used, 3) if neither one-to-one mapping nor active role is present, then many-to-one mapping, 4) if none of the above is used then Adaptive Server Enterprise login and password.
- You can assign external logins to Adaptive Server roles. You can assign anyone with a particular role a corresponding login name and password for any given remote server.
- When you establish a connection to a remote server for a user that has more than one role active, each role is searched for an external login mapping and uses the first mapping it finds to establish the login. This is the same order as displayed by the stored procedure sp_activeroles.
- If you perform role mapping, and a user's role is changed (using set role), any connections made to remote servers that used role mapping must be disconnected. You cannot do this if a transaction is pending. You cannot use set role if a transaction is active and remote connections are present that used role mapping.
- Before running sp_addexternlogin, add the remote server to Adaptive Server with sp_addserver.
- *remote_name* and *remote_password* must be a valid user and password combination on the node where the *server* runs.
- Sites with automatic password expiration need to plan for periodic updates of passwords for external logins.

- Use sp_dropexternlogin to remove the definition of the external login.
- sp_addexternlogin cannot be used from within a transaction.
- The "sa" account and the *login_name* account are the only users who can modify remote access for a given local user.

Permissions

Only the *login_name*, a System Administrator, and a System Security Officer can execute sp_addexternlogin.

See also

System procedures sp_addserver, sp_addserver, sp_helpexternlogin, sp_helpserver

sp_addgroup

Description

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

Creates a group named accounting in the current database:

```
sp_addgroup accounting
```

Usage

- 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).
- A group and a user cannot have the same name.
- Once a group has been created, add new users with sp_adduser. To add an 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.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute sp_addgroup.

See also

Commands grant, revoke

System procedures sp_adduser, sp_changegroup, sp_dropgroup, sp_helpgroup

sp_addlanguage

Description

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—"français", 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 9 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, smalldatetime, date or time 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

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:

```
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
```

Usage

- Usually, you add alternate languages from one of Adaptive Server’s Language Modules using the langinstall utility or the Adaptive Server 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.
- Use sp_modifylogin to change a user’s default language. If you set a user’s default language to a language added with sp_addlanguage, and there are no localization files for the language, the users receive an informational message when they log in, indicating that their client software could not open the localization files.

System Table Changes

- 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 U.S. English.
- The *language* parameter becomes the official language name, stored in the name column of master.dbo.syslanguages. Language names must be unique. Use sp_helplanguage to display a list of the alternate languages available on Adaptive Server.
- sp_addlanguage sets the alias column in master.dbo.syslanguages to the official language name if NULL is entered for alias, but System Administrators can change the value of syslanguage.alias with sp_setlangalias.
- sp_addlanguage sets the upgrade column in master.dbo.syslanguages to 0.

Dates for Languages added with *sp_addlanguage*

- For alternate languages added with Language Modules, Adaptive Server sends date values to clients as datetime 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, where *pubdate* is datetime data and *table* is any table:

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

Permissions

Only a System Administrator can execute sp_addlanguage.

See also

Commands set

System procedures sp_droplanguage, sp_helplanguage, sp_modifylogin, sp_setlangalias

Utilities langinstall

sp_addlogin

Description

Adds a new user account to Adaptive Server; specifies the password expiration interval, the minimum password length, and the maximum number of failed logins allowed for a specified login at creation.

Syntax

sp_addlogin *loginame*, *passwd* [, *defdb*]
    [, *deflanguage*] [, *fullname*] [, *passwdexp*]
    [, *minpwdlen*] [, *maxfailedlogins*]

Parameters

loginame
is the user's login name. Login names must conform to the rules for identifiers.

passwd
is the user's password. Passwords must be at least 6 characters long. If you specify a shorter password, sp_addlogin returns an error message and exits. 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 quotation marks.

defdb
is the name of the default database assigned when a user logs into Adaptive Server. If you do not specify *defdb*, the default, master, is used.

deflanguage
is the official name of the default language assigned when a user logs into Adaptive Server. The Adaptive Server default language, defined by the default language id configuration parameter, 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.

passwdexp
specifies the password expiration interval in days. It can be any value between 0 and 32767, inclusive.

minpwdlen
specifies the minimum password length required for that login. The values range between 0 and 30 characters.

maxfailedlogins
is the number of allowable failed login attempts. It can be any whole number between 0 and 32767.

Examples

Example 1 Creates an Adaptive Server login for “albert” with the password “longer1” and the default database corporate:

```
sp_addlogin albert, longer1, corporate
```

Example 2 Creates an Adaptive Server login for “claire”. Her password is “bleurouge”, her default database is public_db, and her default language is French:

```
sp_addlogin claire, bleurouge, public_db, french
```

Example 3 Creates an Adaptive Server login for “robertw”. His password is “terrible2”, his default database is public_db, and his full name is “Robert Willis”. Do not enclose null in quotes:

```
sp_addlogin robertw, terrible2, public_db, null,
"Robert Willis"
```

Example 4 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. Do not enclose null in quotes:

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

Example 5 An alternative way of creating the login shown in example 4:

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

Usage

- For ease of management, it is strongly recommended that all users’ Adaptive Server login names be the same as their operating system login names. This makes it easier to correlate audit data between the operating system and Adaptive Server. Otherwise, keep a record of the correspondence between operating system and server 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*.

Permissions

Only a System Security Officer can execute sp_addlogin.

See also

System procedures sp_addalias, sp_adduser, sp_droplogin, sp_locklogin, sp_modifylogin, sp_password, sp_role

sp_addmessage

Description

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* [, *with_log* [, replace]]]

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 1024 bytes.

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

with_log
specifies whether the message is logged in the Adaptive Server error log as well as in the Windows NT Event Log on Windows NT servers, if logging is enabled. If *with_log* is TRUE, the message is logged, regardless of the severity of the error. If *with_log* is FALSE, the message may or may not be logged, depending on the severity of the error. If you do not specify a value for *with_log*, the default is FALSE.

replace
specifies whether to overwrite an existing message of the same number and *languid*. If replace is specified, the existing message is overwritten; if replace is omitted, it is not. If you do not specify a value for replace, the parameter's default behavior specifies that the existing message will not be overwritten.

Examples

Example 1 Adds a message with the number 20001 to sysusermessages:

```
sp_addmessage 20001, "The table '%1!' is not owned by
the user '%2!'."
```

Example 2 Adds a message with the number 20002 to sysusermessages. This message is logged in the Adaptive Server error log, as well as in the Windows NT Event Log on Windows NT servers, if event logging is enabled. If a message numbered 20002 exists in the default session language, this message overwrites the old message:

```
sp_addmessage 20002, "The procedure'%1!' is not owned
```

```
by the user '%2!'.", NULL, TRUE, "replace"
```

Usage

- sp_addmessage does not overwrite an existing message of the same number and *langid* unless you specify @replace = "replace".
- print and raiserror recognize placeholders in the message text to print out. A single message can contain up to 20 unique placeholders in any order. These placeholders 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 placeholders are numbered to allow reordering of the arguments when Adaptive Server is translating a message to a language with a different grammatical structure. A placeholder 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.

Permissions

Any user can execute sp_addmessage.

See also

Commands print, raiserror

System procedures sp_altermessage, sp_bindmsg, sp_dropmessage, sp_getmessage

sp_addobjectdef

Description

Component Integration Services only Specifies the mapping between a local table and an external storage location.

Syntax

sp_addobjectdef *tablename*, "*objectdef*" [,"*objecttype*"]

Parameters

tablename

is the name of the object as it is defined in a local table. The *tablename* can be in any of the following forms:

dbname.*owner*.*object*
dbname..*object*
owner.*object*
object

dbname and *owner* are optional. *object* is required. If you do not specify an *owner*, the default (current user name) is used. If you specify a *dbname*, it must be the current database name, and you must specify *owner* or mark the owner with a placeholder in the format *dbname*..*object*. Enclose any multipart *tablename* values in quotes.

objectdef

is a string naming the external storage location of the object. The *objecttype* at *objectdef* can be a table, view, or read-only remote procedure call (RPC) result set accessible to a remote server. A table, view, or RPC uses the following format for *objectdef*:

```
server_name.dbname.owner.object
```

server_name and *object* are required. *dbname* and *owner* are optional, but if they are not supplied, a placeholder in the format *dbname*..*object*, is required.

See "Server Classes" in the *Component Integration Services User's Guide* for more information.

objecttype

is one of the values that specify the format of the object named by *objectdef*. Table 1-2 describes the valid values. Enclose the *objecttype* value in quotes.

Table 1-2: Allowable values for objecttype

Value	Description
table	Indicates that the object named by *objectdef* is a table accessible to a remote server. This value is the default for *objecttype*.
view	Indicates that the object named by *objectdef* is a view managed by a remote server and processed as a table.

Value	**Description**
rpc	Indicates that the object named by *objectdef* is an RPC managed by a remote server. Adaptive Server processes the result set from the RPC as a read-only table.

Table 1-3 summarizes how each *objecttype* is used.

Table 1-3: Summary of objecttype uses

objecttype	***create table***	***create existing table***	**Write to table**	**Read from table**
table	Yes	Yes	Yes	Yes
view	No	Yes	Yes	Yes
rpc	No	Yes	No	Yes

Examples

Example 1 Maps the local table accounts in the database finance to the remote object pubs.dbo.accounts in the remote server named SYBASE. The current database must be finance. A subsequent create table creates a table in the pubs database. If pubs.dbo.accounts is an existing table, a create existing table statement populates the table finance.dbo.accounts with information about the remote table:

```
sp_addobjectdef "finance.dbo.accounts",
"SYBASE.pubs.dbo.accounts", "table"
```

Example 2 Maps the local table stockcheck to an RPC named stockcheck on remote server NEWYORK in the database wallstreet with owner “kelly”. The result set from RPC stockcheck is seen as a read-only table. Typically, the next operation would be a create existing table statement for the object stockcheck:

```
sp_addobjectdef stockcheck,
"NEWYORK.wallstreet.kelly.stockcheck", "rpc"
```

Usage

- sp_addobjectdef specifies the mapping between a local table and an external storage location. It identifies the format of the object at that location.You can use sp_addobjectdef only when Component Integration Services is installed and configured.
- sp_addobjectdef replaces the sp_addtabledef command. sp_addobjectdef allows existing scripts to run without modification. Internally, sp_addtabledef invokes sp_addobjectdef.
- Only the System Administrator can provide the name of another user as a table owner.
- When *objecttype* is table, view, or rpc, the *objectdef* parameter takes the following form:

```
"server_name.database.owner.tablename"
```

- *server_name* – represents a server that has already been added to sysservers by sp_addserver.
- *database* – may not be required. Some server classes do not support it.
- *owner* – should always be provided, to avoid ambiguity. If you do not specify *owner*, the remote object referenced may vary, depending on whether or not the external login corresponds to the remote object owner.
- *tablename* – is the name of a remote server table.

- Use sp_addobjectdef before issuing any create table or create existing table commands. create table is valid only for the *objecttype* values table and file. When either create table or create existing table is used, Adaptive Server checks sysattributes to determine whether any table mapping has been specified for the object. Follow the *objecttype* values view and rpc with create existing table statements.
- After the table has been created, all future references to the local table name (by select, insert, delete and update) are mapped to the correct location.
- For information about RMS, see the Component Integration Services User's Guide.

Permissions

Any user can execute sp_addobjectdef.

See also

Commands create existing table, create table, drop table

System procedures sp_addlogin, sp_addserver, sp_defaultloc, sp_dropobjectdef, sp_helpserver

sp_add_qpgroup

Description	Adds an abstract plan group.
Syntax	sp_add_qpgroup *new_name*
Parameters	*new_name* is the name of the new abstract plan group. Group names must be valid identifiers.
Examples	Creates a new abstract plan group named dev_plans:

```
sp_add_qpgroup dev_plans
```

Usage

- Use sp_add_qpgroup to add abstract plan groups for use in capturing or creating abstract plans. The abstract plan group must exist before you can create, save, or copy plans into a group.
- sp_add_qpgroup cannot be run in a transaction.

Permissions: Only a System Administrator or Database Owner can execute sp_add_qpgroup.

See also

Commands set

System procedures sp_help_qpgroup

sp_addremotelogin

Description

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

Syntax

sp_addremotelogin *remoteserver* [, *loginame* [, *remotename*]]

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, which was created with sp_addserver.

Note This manual page uses the term "local server" to refer to the server that is executing the remote procedures run from a "remote server."

---

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

remotename
is the name used by the remote server when logging into the local server. All *remotenames* that are not explicitly matched to a local *loginame* are automatically matched to a local name. In Example 1 , the local name is the remote name that is used to log in. In Example 2 , the local name is "albert."

Examples

Example 1 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 to local names when the local and remote servers have the same users:

```
sp_addremotelogin GATEWAY
```

This example results in a value of -1 for the suid column and a value of NULL for the remoteusername in a row of sysremotelogins.

Example 2 Creates an entry that maps all logins from the remote server GATEWAY to the local user name "albert". Adaptive Server adds a row to sysremotelogins with Albert's server user ID in the suid column and a null value for the remoteusername:

```
sp_addremotelogin GATEWAY, albert
```

For these logins to be able to run RPCs on the local server, they must specify a password for the RPC connection when they log into the local server, or they must be "trusted" on the local server. To define these logins as "trusted", use sp_remoteoption.

Example 3 Maps a remote login from the remote user “pogo” on the remote server GATEWAY to the local user “ralph”. Adaptive Server adds a row to sysremotelogins with Ralph’s server user ID in the suid column and “pogo” in the remoteusername column:

```
sp_addremotelogin GATEWAY, ralph, pogo
```

Usage

- When a remote login is received, the local server tries to map the remote user to a local user in three different ways:
    - First, the local server looks for a row in sysremotelogins that matches the remote server name and the remote user name. If the local server finds a matching row, the local server user ID for that row is used to log in the remote user. This applies to mappings from a specified remote user.
    - If no matching row is found, the local server searches for a row that has a null remote name and a local server user ID other than -1. If such a row is found, the remote user is mapped to the local server user ID in that row. This applies to mappings from any remote user from the remote server to a specific local name.
    - Finally, if the previous attempts failed, the local server checks the sysremotelogins table for an entry that has a null remote name and a local server user ID of -1. If such a row is found, the local server uses the remote name supplied by the remote server to look for a local server user ID in the syslogins table. This applies when login names from the remote server and the local server are the same.
- The name of the local user may be different on the remote server.
- If you use sp_addremotelogin to map all users from a remote server to the same local name, use sp_remoteoption to specify the “trusted” option for those users. For example, if all users from the server GOODSRV that are mapped to “albert” are to be “trusted”, use sp_remoteoption as follows:

```
sp_remoteoption GOODSRV, albert, NULL, trusted, true
```

  Logins that are not specified as “trusted” cannot execute RPCs on the local server unless they specify passwords for the local server when they log into the remote server. In Open Client™ Client-Library™, the user can use the ct_remote_pwd routine to specify a password for server-to-server connections. isql and bcp do not permit users to specify a password for RPC connections.

If users are logged into the remote server using “unified login”, these logins are already authenticated by a security mechanism. These logins must also be trusted on the local server, or the users must specify passwords for the server when they log into the remote server.

- Every remote login entry has a status. The default status for the trusted option is false (not trusted). This means that when a remote login comes in using that entry, the password is checked. If you do not want the password to be checked, change the status of the trusted option to true with sp_remoteoption.

Permissions

Only a System Administrator can execute sp_addremotelogin.

See also

Documents See the *System Administration Guide* for more information about setting up servers for remote procedure calls and for using “unified login.”

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

Utility isql

sp_add_resource_limit

Description

Creates a limit on the number of server resources that can be used by an Adaptive Server login and/or an application to execute a query, query batch, or transaction.

Syntax

sp_add_resource_limit *name*, *appname*, *rangename, limittype*, *limitvalue*
    [, *enforced* [, *action* [, *scope*]]]

Parameters

name

is the Adaptive Server login to which the limit applies. You must specify either a *name* or an *appname* or both. To create a limit that applies to all users of a particular application, specify a *name* of NULL.

appname

is the name of the application to which the limit applies. You must specify either a *name* or an *appname* or both. To create a limit that applies to all applications used by an Adaptive Server login, specify an *appnam*e of null. To create a limit that applies to a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename

is the time range during which the limit is enforced. The time range must exist in the systimeranges system table of the master database at the time you create the limit.

limittype

is the type of resource to limit. This must be one of the following:

Limit type	**Description**
row_count	Limits the number of rows a query can return
elapsed_time	Limits the number of seconds, in wall-clock time, that a query batch or transaction can run
io_cost	Limits either the actual cost or the optimizer’s cost estimate for processing a query
tempdb_space	Limits the number of pages a tempdb database can have during a single session

limitvalue

is the maximum amount of the server resource (I/O cost, elapsed time in seconds, row count, or tempdb space) that can be used by the login or application before Adaptive Server enforces the limit. This must be a positive, nonzero integer that is less than or equal to 2^{31}. The following table indicates what value to specify for each limit type:

Limit type	**Limit value**
row_count	The maximum number of rows that can be returned by a query before the limit is enforced.

Limit type	Limit value
elapsed_time	The number of seconds, in wall-clock time, that a query batch or transaction can run before the limit is enforced.
io_cost	A unitless measure derived from the optimizer's costing formula.
tempdb_space	The number of pages used in tempdb per session.

enforced

determines whether the limit is enforced prior to or during query execution. The following table lists the valid values for each limit type:

***enforced* code**	**Description**	**Limit type**
1	Action is taken when the estimated I/O cost of execution exceeds the specified limit.	io_cost
2	Action is taken when the actual row count, elapsed time, or I/O cost of execution exceeds the specified limit.	row_count elapsed_time io_cost
3	Action is taken when either the estimated cost or the actual cost exceeds the specified limit.	io_cost

If you specify an *enforced* value of 3, Adaptive Server performs a logical "or" of 1 and 2. For example, assume *enforced* is set to 3. If you run a query whose io_cost exceeds the estimated cost, the specified *action* is executed. If the query is within the limits specified for estimated cost but exceeds the actual cost, the specified *action* is also executed.

If you do not specify an *enforced* value, Adaptive Server enforces limit 2 for row_count and elapsed_time and limit 3 for io_cost. In other words, if the limit type is io_cost, the specified action is executed if the query exceeds either the estimated or actual cost.

action

is the action to take when the limit is exceeded. The following action codes are valid for all limit types:

***action* code**	**Description**
1	Issues a warning
2	Aborts the query batch
3	Aborts the transaction
4	Kills the session

If you do not specify an *action* value, Adaptive Server uses a default value of 2 (abort the query batch).

scope

is the scope of the limit. Specify one of the following codes appropriate to the type of limit:

***scope* code**	**Description**	**Limit type**
1	Query	io_cost row_count
2	Query batch (one or more SQL statements sent by the client to the server)	elapsed_time
4	Transaction	elapsed_time
6	Query batch *and* transaction	elapsed_time

If you do not specify a *scope* value, the limit applies to all possible scopes for the limit type.

Examples

Example 1 Creates a resource limit that applies to all users of the payroll application during the early_morning time range. If the query batch takes more than 120 seconds to execute, Adaptive Server issues a warning:

```
sp_add_resource_limit NULL, payroll, early_morning, elapsed_time, 120, 2,
    1, 2
```

Example 2 Creates a resource limit that applies to all ad hoc queries and applications run by “joe_user” during the midday time range. When a query returns more than 5000 rows, Adaptive Server aborts the transaction:

```
sp_add_resource_limit joe_user, NULL, midday, row_count, 5000, 2, 3, 1
```

Example 3 Creates a resource limit that applies to all ad hoc queries and applications run by “joe_user” during the midday time range. When the optimizer estimates that the I/O cost would exceed 650, Adaptive Server aborts the transaction:

```
sp_add_resource_limit joe_user, NULL, midday, io_cost, 650, 1, 3, 1
```

Usage

- You must enable sp_configure "allow resource limits" for resource limits to take effect.
- Multiple resource limits can exist for a given user, application, limit type, scope, and enforcement time, as long as their time ranges do not overlap.
- All limits for the currently active named time ranges and the “at all times” range for a login and/or application name are bound to the user’s session at login time. Therefore, if a user logs into Adaptive Server independently of a given application, resource limits that restrict the user in combination with that application do not apply. To guarantee restrictions on that user, create a resource limit that is specific to the user and independent of any application.

- Since either the user login name or application name, or both, are used to identify a resource limit, Adaptive Server observes a predefined search precedence while scanning the sysresourcelimits table for applicable limits for a login session. The following table describes the precedence of matching ordered pairs of login name and application name:

Level	Login name	Application name
1	"joe_user"	payroll
2	NULL	payroll
3	"joe_user"	NULL

  If one or more matches are found for a given precedence level, no further levels are searched. This prevents conflicts regarding similar limits for different login/application combinations.

  If no match is found at any level, no limit is imposed on the session.

- When you add, delete, or modify resource limits, Adaptive Server rebinds the limits for each session for that login and/or application at the beginning of the next query batch for that session.
- When you change the currently active time ranges, Adaptive Server rebinds limits for the session. This rebinding occurs at the beginning of the next query batch.
- You cannot associate the limits for a particular login, application, or login/application combination with named time ranges that overlap (except for limits that share the same time range).

  For example, if a user is limited to retrieving 50 rows between 9:00 a.m. and 1:00 p.m., you cannot create a second resource limit for the same user that limits him to retrieving 100 rows between 10:00 a.m. and 12:00 noon. However, you can create a resource hierarchy by assigning the 100-row limit to the *user* between 10:00 a.m. and 12:00 noon and assigning the 50-row limit to an *application*, like isql, between 9:00 a.m. and 1:00 p.m.

Note Although Adaptive Server terminates the current transaction when it reaches its time limit, you receive no 1105 error message until you issue another SQL command or batch; in other words, the message appears only when you attempt to use the connection again.

Permissions

Only a System Administrator can execute sp_add_resource_limit.

See also

Documents For more information on resource limits, see the *System Administration Guide*.

System procedures sp_configure, sp_drop_resource_limit, sp_help_resource_limit, sp_modify_resource_limit

Utility isql

sp_addsegment

Description

Defines a segment on a database device in a 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 in which to define the segment. *dbname* must be the name of the current database or match the database name qualifying sp_addsegment.

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

Examples

Example 1 Creates a segment named indexes for the database pubs2 on the database device named dev1:

```
sp_addsegment indexes, pubs2, dev1
```

Example 2 Creates a segment named indexes for the database pubs2 on the database device named dev1:

```
disk init
    name = "pubs2_dev",
    physname = "/dev/pubs_2_dev",
    vdevno = 9, size = 5120
go
alter database pubs2 on pubs2_dev = 2
go
pubs2..sp_addsegment indexes, pubs2, dev1
```

Usage

- sp_addsegment defines segment names for database devices created with disk init and 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 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 you place it on a separate device with the log on extension to create database or with sp_logdevice. See the *System Administration Guide* for more information.
- Although you can use sp_addsegment in a database that has both data and the log on the same device, such as when the database is created without the log on option, Adaptive Server returns an error message if you create a database using:

```
create database dbname on devicename log on
devicename with override
```

Permissions

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

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

Description

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

Syntax

sp_addserver "*lname*" [, *class* [," *pname*"]]

Parameters

"*lname*"

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

class

identifies the category of server being added. A server *class* of "null" defaults to "ASEnterprise". Table 1-4 lists allowable values for the *class* parameter:

Table 1-4: Allowable values for server_class parameter

***class* parameter value**	**Description**
local	Local server (there can be only one) used only once after start-up, or after restarting Adaptive Server, to identify the local server name so that it can appear in messages printed by Adaptive Server
null	Remote server with no category defined
ASEnterprise	All versions of Adaptive Server Enterprise; support for SQL Server 4.9 is not provided.
ASAnywhere	Adaptive Server Anywhere version 6.0 or later.
ASIQ	A server with server class ASIQ is any version of Adaptive Server IQ of 12.0 or later.
db2	**Component Integration Services only** An IBM DB2 database accessed through: • DirectConnect for MVS / TRS (can also be configured as server class direct_connect) • Direct (gateway less) access to Mainframe Connect
direct_connect	**Component Integration Services only** An Open Server-based application that conforms to the direct_connect interface specification.
sds	Conforms to the interface requirements of a Specialty Data Store™ as described in the Adaptive Server Specialty Data Store Developer's Kit manual.

See "Remote Servers" in Chapter 2, "Understanding Component Integration Services" of the *Component Integration Services User's Guide* for details on these parameter values.

pname

is the name in the interfaces file for the server named *lname*. This enables you to establish local aliases for other Adaptive Servers or Backup Servers that you may need to communicate with. If you do not specify a *pname*, *lname* is used.

Component Integration Services only – You can use *pname* to specify the hostname or IP address and the port of the server you wish to connect to. This enables you to bypass the need for directory services (such as LDAP or an interfaces file) for the server when using the CT-Library. Use the following format:

```
"hostname:port"

"ipaddr:port"
```

Note You must enclose the hostname and port with single or double quotes to use this option.

Examples

Example 1 Adds an entry for a remote server named GATEWAY in master.dbo.sysservers. The *pname* is also GATEWAY:

```
sp_addserver GATEWAY
```

Example 2 Adds an entry for a remote server named GATEWAY in master.dbo.sysservers. The *pname* is VIOLET. If there is already a sysservers entry for GATEWAY with a different *pname*, the *pname* of server GATEWAY changes to VIOLET:

```
sp_addserver GATEWAY, null, VIOLET
```

Example 3 Adds an entry for the local server named PRODUCTION:

```
sp_addserver PRODUCTION, local
```

Example 4 *Component Integration Services only* – Adds an entry for a remote Adaptive Server with the host name "myhost" with port number 10224:

```
sp_addserver S1, ASEnterprise, "myhost:10224"
```

Note If you use this syntax for *pname*, the Adaptive Server site handler cannot successfully connect to this server; only CIS connections recognize this syntax for *pname*.

Example 5 *Component Integration Services only* – Adds an entry for a remote Adaptive Server with the host IP 192.123.456.010 with port number 11222:

```
sp_addserver S3, direct_connect, "192.123.456.010:11222"
```

Usage

- 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, the remote server must exist in the sysservers table.
- If *lname* already exists as a server name in the sysservers table, sp_addserver changes the remote server's srvnetname to the name specified by *pname*. When it does this, sp_addserver reports 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.
- Adaptive Server requires that the Backup Server have an *lname* 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 *pname* for server SYB_BACKUP with sp_addserver so that Adaptive Server can communicate with Backup Server for database dumps and loads.
- If you specify an *lname*, *pname* and *class* 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.

Permissions

Only a System Security Officer can execute sp_addserver.

See also

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

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

sp_addthreshold

Description

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

Syntax

sp_addthreshold *dbname*, *segname*, *free_space*, *proc_name*

Parameters

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

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

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

proc_name
is the stored procedure to be executed when the amount of free space on *segname* drops below *free_space*. The procedure can be located in any database on the current Adaptive Server or on an Open Server. Thresholds cannot execute procedures on remote Adaptive Servers.

Examples

Example 1 Creates a threshold for segment1. When the free space on segment1 drops below 200 pages, Adaptive Server executes the procedure pr_warning:

```
sp_addthreshold mydb, segment1, 200, pr_warning
```

Example 2 Creates a threshold for the user_data segment. When the free space on user_data falls below 100 pages, Adaptive Server executes a remote procedure call to the Open Server mail_me procedure:

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

Example 3 Creates a threshold on the indexes segment of the pubs2 database. You can issue this command from any database:

```
pubs2..sp_addthreshold pubs2, indexes, 100, pr_warning
```

Usage

Crossing a threshold

- When a threshold is crossed, Adaptive Server executes the associated stored procedure. Adaptive Server uses the following search path for the threshold procedure:

    - If the procedure name does not specify a database, Adaptive 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_", Adaptive Server looks in the sybsystemprocs database.

  If the procedure is not found in either database, Adaptive Server sends an error message to the error log.

- Adaptive 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, Adaptive 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, called the *last-chance threshold,* is calculated by Adaptive Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, Adaptive Server suspends the transaction until log space is freed. Use sp_dboption to change this behavior for a particular database. sp_dboption "abort tran on log full", true causes Adaptive Server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.
- 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.

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 sybsystemprocs database, and the 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.
- sp_addthreshold checks to ensure that the user adding the threshold procedure has been directly granted the "sa_role". All system roles active when the threshold procedure is created are entered in systhresholds as valid roles for the user writing the procedure. However, only directly granted system roles are activated when the threshold fires. Indirectly granted system roles and user-defined roles are not activated.
- Adaptive 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 it 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 is executed for all thresholds in Adaptive Server.
- 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".

- Adaptive Server executes the threshold procedure with the permissions the user had at the time he or she added the threshold, minus any permissions that have since been revoked.
- Each threshold procedure uses one user connection, for as long as it takes for the procedure to execute.

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, Adaptive Server executes the threshold with the permissions the owner had 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

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

- 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.

Permissions

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

See also

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

Commands create procedure, dump transaction

Functions lct_admin

System procedures sp_dboption, sp_dropthreshold, sp_helpthreshold, sp_modifythreshold, sp_thresholdaction

sp_add_time_range

Description

Adds a named time range to an Adaptive Server.

Syntax

sp_add_time_range *name*, *startday*, *endday*,
    starttime, *endtime*

Parameters

name

is the name of the time range. Time range names must be 30 characters or fewer. The name cannot already exist in the systimeranges system table of the master database.

startday

is the day of the week on which the time range begins. This must be the full weekday name for the default server language, as stored in the syslanguages system table of the master database.

endday

is the day of the week on which the time range ends. This must be the full weekday name for the default server language, as stored in the syslanguages system table of the master database. The *endday* can fall either earlier or later in the week than the *startday* or can be the same day as the *startday*.

starttime

is the time of day when the time range begins. Specify the *starttime* in terms of a 24-hour clock, with a value between “00:00” (midnight) and “23:59” (11:59 p.m.). Use the following form:

```
"HH:MM"
```

endtime

is the time of day when the time range ends. Specify the *endtime* in terms of a 24-hour clock, with a value between “00:00” (midnight) and “23:59” (11:59 p.m.). Use the following form:

```
"HH:MM"
```

Note To create a time range that spans the entire day, specify both a start time and an end time of “00:00”.

The *endtime* must occur later in the day than the *starttime,* unless *endtime* is “00:00”.

Examples

Example 1 Creates the business_hours time range, which is active Monday through Friday, from 9:00 a.m. to 5:00 p.m.:

```
sp_add_time_range business_hours, monday, Friday, "09:00", "17:00"
```

Example 2 Creates two time ranges, before_hours and after_hours, that, together, span all non-business hours Monday through Friday. The before_hours time range covers the period from 12:00 midnight to 9:00 a.m., Monday through Friday. The after_hours time range covers the period from 6:00 p.m. through 12:00 midnight, Monday through Friday:

```
sp_add_time_range before_hours, Monday, Friday, "00:00", "09:00"
```

```
sp_add_time_range after_hours, Monday, Friday, "18:00", "00:00"
```

Example 3 Creates the weekends time range, which is 12:00 midnight Saturday to 12:00 midnight Sunday:

```
sp_add_time_range weekends, Saturday, Sunday, "00:00", "00:00"
```

Example 4 Creates the Fri_thru_Mon time range, which is 9:00 a.m. to 5:00 p.m., Friday, Saturday, Sunday, and Monday:

```
sp_add_time_range Fri_thru_Mon, Friday, Monday, "09:00", "17:00"
```

Example 5 Creates the Wednesday_night time range, which is Wednesday from 5:00 p.m. to 12:00 midnight:

```
sp_add_time_range Wednesday_night, Wednesday, Wednesday, "17:00", "00:00"
```

Usage

- Adaptive Server includes one named time range, the “at all times” time range. This time range covers all times, from the first day through the last of the week, from 00:00 through 23:59. It cannot be modified or deleted.
- Adaptive Server generates a unique ID number for each named time range and inserts it into the systimeranges system table,
- When storing a time range in the systimeranges system table, Adaptive Server converts its *startday* and *endday* values into integers. For servers with a default language of us_english, the week begins on Monday (day 1) and ends on Sunday (day 7).
- It is possible to create a time range that overlaps with one or more other time ranges.
- Range days are contiguous, so the days of the week can wrap around the end to the beginning of the week. In other words, Sunday and Monday are contiguous days, as are Tuesday and Wednesday.

- The active time ranges are bound to a session at the beginning of each query batch. A change in the server's active time ranges due to a change in actual time has no effect on a session during the processing of a query batch. In other words, if a resource limit restricts a query batch during a given time range but a query batch begins before that time range becomes active, the query batch that is already running is not affected by the resource limit.
- The addition, modification, and deletion of time ranges using the system procedures does not affect the active time ranges for sessions currently in progress.
- If a resource limit has a transaction as its scope, and a change occurs in the server's active time ranges while a transaction is running, the newly active time range does not affect the transaction currently in progress.
- Changes to a resource limit that has a transaction as its scope does not affect any transactions currently in progress.
- For more information on time ranges, see the System Administration Guide.

Permissions

Only a System Administrator can execute sp_add_time_range.

See also

System procedures sp_add_resource_limit, sp_drop_time_range, sp_modify_time_range

sp_addtype

Description

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 in each database.

phystype

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

The char, varchar, unichar, univarchar, nchar, nvarchar, binary, and varbinary datatypes expect a *length* in parentheses. If you do not supply one, Adaptive Server uses the default length of 1 character.

The float datatype expects a binary *precision* in parentheses. If you do not supply one, Adaptive 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, Adaptive Server uses a default precision of 18 and a 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 by Adaptive Server. The value of the IDENTITY column uniquely identifies each row in a table. IDENTITY columns are not updatable and do not allow null values.

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". Any *nulltype* that includes a blank space must be enclosed in single or double quotes.

If you omit both the IDENTITY property and the *nulltype*, Adaptive 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 compliance to the SQL standards, 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

Example 1 Creates a user-defined datatype called ssn to be used for columns that hold social security numbers. Since the *nulltype* parameter is not specified, Adaptive 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):

```
sp_addtype ssn, "varchar(11)"
```

Example 2 Creates a user-defined datatype called birthday that allows null values:

```
sp_addtype birthday, "datetime", null
```

Example 3 Creates a user-defined datatype called temp52 used to store temperatures of up to 5 significant digits with 2 places to the right of the decimal point:

```
sp_addtype temp52, "numeric(5,2)"
```

Example 4 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 of up to 10 digits in length:

```
sp_addtype "row_id", "numeric(10,0)", "identity"
```

Example 5 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:

```
sp_addtype systype, sysname
```

Usage

- 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 Adaptive Server-supplied datatypes, specifying the length or the 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*, 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.
- If unichar or univarchar is specified as the *phystype*, the maximum length of columns created with the new type is the length specified in sp_addtype multiplied by the value of 2 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
  ```

Datatypes 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 IDENTITY, NOT NULL, or neither 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 null type 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.

Permissions

Any user can execute sp_addtype.

See also

Commands create default, create rule, create table

Datatypes User-defined datatypes

System procedures sp_bindefault, sp_bindrule, sp_dboption, sp_droptype, sp_rename, sp_unbindefault, sp_unbindrule

sp_addumpdevice

Description

Adds a dump device to Adaptive Server.

Syntax

sp_addumpdevice {"tape" | "disk"}, *logicalname*,
    physicalname [, *tapesize*]

Parameters

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

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

logicalname
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 in the load and dump commands.

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

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

Examples

Example 1 Adds a 40MB tape device. Dump and load commands can reference the device by its physical name, */dev/nrmt8*, or its logical name, mytapedump:

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

Example 2 Adds a disk device named mydiskdump. Specify an absolute or relative path name and a file name:

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

Usage

- sp_addumpdevice adds a dump device to the master.dbo.sysdevices table. Tape devices are assigned a cntrltype of 3; disk devices are assigned 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 *tapesize* parameter. If you specify a relative path name, dumps are made to—or loaded from—the current Adaptive Server working directory at the time the dump or load command executes.
- 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 *tapesize* 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.

Permissions

Only a System Administrator can execute sp_addumpdevice.

See also

Commands disk init, dump database, dump transaction, load database, load transaction

System procedures sp_dropdevice, sp_helpdevice

sp_adduser

Description	Adds a new user to the current database.
Syntax	sp_adduser *loginame* [, *name_in_db* [, *grpname*]]

Parameters

loginame
is the user's name 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

Example 1 Adds "margaret" to the database. Her database user name is the same as her Adaptive Server login name, and she belongs to the default group, "public":

```
sp_adduser margaret
```

Example 2 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 to the default group "public":

```
sp_adduser haroldq, harold, fort_mudge
```

Usage

- 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 that is different from his or her login name in Adaptive 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, since it is not mapped to the identity and privileges of another user.
- A user and a group cannot have the same name.
- 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.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute sp_adduser.

See also

Commands grant, revoke, use

System procedures sp_addalias, sp_addgroup, sp_changegroup, sp_dropalias, sp_dropgroup, sp_helpuser

sp_altermessage

Description

Enables and disables the logging of a system-defined or user-defined message in the Adaptive Server error log.

Syntax

sp_altermessage *message_id*, *parameter, parameter_value*

Parameters

message_id
is the message number of the message to be altered. This is the number of the message as it is recorded in the error column in the sysmessages or sysusermessages system table.

parameter
is the message parameter to be altered. The maximum length is 30 bytes. The only valid parameter is with_log.

parameter_value
is the new value for the parameter specified in *parameter*. The maximum length is 5 bytes. Values are true and false.

Examples

Specifies that message number 2000 in sysmessages should be logged in the Adaptive Server error log and also in the Windows NT Event Log (if logging is enabled):

```
sp_altermessage 2000, 'with_log', 'TRUE'
```

Usage

- If the *parameter_value* is true, the specified message is always logged. If it is false, the default logging behavior is used; the message may or may not be logged, depending on the severity of the error and other factors. Setting the *parameter_value* to false produces the same behavior that would occur if sp_altermessage had not been called.
- On Windows NT servers, sp_altermessage also enables and disables logging in the Windows NT Event Log.

Permissions

Only the Database Owner or a System Administrator can execute sp_altermessage.

See also

System procedures sp_addmessage, sp_dropmessage

sp_audit

Description — Allows a System Security Officer to configure auditing options.

Syntax — sp_audit *option*, *login_name*, *object_name* [,*setting*]

Parameters

option
is the name of the auditing option to set. Table 1-5 lists the valid auditing options.

Table 1-5: Auditing options

Option	Description
adhoc	Allows users to use sp_addauditrecord to add their own user-defined audit records to the audit trail.
all	Audits all actions performed by a particular user or by users with a particular role. You can only use this option to specify system roles. **Note** Auditing all actions does not affect whether users can add ad hoc audit records.
alter	Audits the execution of the alter table or alter database commands.
bcp	Audits the execution of the bcp in utility.
bind	Audits the execution of sp_bindefault, sp_bindmsg, and sp_bindrule system procedures.
cmdtext	Audits all actions of a particular user.
create	Audits the creation of database objects.
dbaccess	Audits access to the current database from another database.
dbcc	Audits the execution of any dbcc command.
delete	Audits the deletion of rows from a table or view.
disk	Audits the execution of disk init, disk refit, disk reinit, disk mirror, disk unmirror, and disk remirror.
drop	Audits the dropping of database objects.
dump	Audits the execution of dump database or dump transaction.
errors	Audits errors, whether fatal or not.
exec_procedure	Audits the execution of a stored procedure.
exec_trigger	Audits the execution of a trigger.
func_dbaccess	Audits access to a database via a Transact-SQL function.
func_obj_access	Audits access to a database object via a Transact-SQL function.
grant	Audits the execution of the grant.
insert	Audits the insertion of rows into a table or view.
install	Audits the installation of Java classes.
load	Audits the execution of the load database or load transaction
login	Audits all login attempts into Adaptive Server.
logout	Audits all logout attempts from Adaptive Server.

Option	Description
mount	Audits mount database commands.
quiesce	Audits quiesce database commands.
reference	Audits references between tables.
remove	Audits the removal of Java classes.
revoke	Audits the execution of the revoke.
rpc	Audits the execution of remote procedure calls.

Option	Description
security	Audits the following security-relevant events: • Starting up or shutting down the server • Activating or deactivating a role • Issuing any of the following commands: • addcert • connect • create and drop login • dropcert • create, drop, alter, grant, and revoke role • kill • online database • set proxy • set session authorization • sp_configure • The following functions: • config_admin • set_password • valid_user • attr_notify • ha_add_companion • ha_remove_companion • ha_check_alive • ha_getversion • ha_failback • ha_retrestrictionclass • ha_getrcs • ha_setrcs • ha_hacluster_verify • js_wakeup • unlock_admin_account • ssl_admin • Using any of the following functions: • valid_user • proc_role (from within a system procedure) • Regenerating the SSO passwords

Option	Description
select	Audits the execution of the select.
setuser	Audits the execution of the setuser.
table_access	Audits access to any table by a specific user.
truncate	Audits the execution of the truncate table.
unbind	Audits the execution of the sp_unbindrule, sp_unbindmsg, and sp_unbindefault.
unmount	Audits the execution of the umount database command.
update	Audits updates to rows in a table or view.
view_access	Audits access to any view by a specific user.

login_name

is the parameter that lets you specify all, a system role, or the name of a specific login to be audited. However, system roles can only be specified if you use the all option. You cannot audit individual options for a system role.

object_name

is the name of the object to be audited. Valid values, depending on the value you specified for *option*, are:

- The object name, including the owner's name if you do not own the object. For example, to audit a table named inventory that is owned by Joe, you would specify joe.inventory for *object_name*.
- all for all objects.
- default table, default view, default procedure, or default trigger to audit access to any new table, view, procedure, or trigger.

  default table and default view are valid values for object_name when you specify delete, insert, select, or update for the option parameter. default procedure is valid when you specify the exec_procedure option. default trigger is valid when you specify the exec_trigger option.

See the *System Administration Guide* for more information about the *object_name* values that are valid with each *option* value.

setting

is the level of auditing. If you do not specify a value for *setting*, Adaptive Server displays the current auditing setting for the option. Valid values for the *setting* parameter are described in the following table:

***setting* value**	**Description**
on	Activates auditing for the specified option. Adaptive Server generates audit records for events controlled by this option, whether the event passes or fails permission checks.
off	Deactivates auditing for the specified option.

***setting* value**	**Description**
pass	Activates auditing for events that pass permission checks.
fail	Deactivates auditing for events that fail permission checks.

If you specify pass for an option and later specify fail for the same option, or vice versa, the result is equivalent to specifying on. Adaptive Server generates audit records regardless of whether events pass or fail permission checks. Settings of on or off apply to all auditing options. Settings of pass and fail apply to all options except errors and adhoc. For these options, only on or off applies. The initial, default value of all options is off.

Examples

Example 1 Initiates auditing for SSL security-relevant events. Both successful and failed events are audited:

```
sp_audit "security", "all", "all", "on"
sample records added:
```

To view the events from sybsecurity:

```
select * from sybsecurity..sysaudits_01 where event=99
```

Example 2 Displays the setting of the security auditing option:

```
sp_audit "security", "all", "all"
```

Example 3 Initiates auditing for the creation of objects in the master database, including create database.

```
sp_audit "create", "all", master, "on"
```

Example 4 Initiates auditing for the creation of all objects in the db1database:

```
sp_audit "create", "all", db1, "on"
```

Example 5 Initiates auditing for all failed executions by a System Administrator.

```
sp_audit "all", "sa_role", "all", "fail"
```

Example 6 Initiates auditing for all updates to future tables in the current database. For example, if the current database is utility, all new tables created in utility will be audited for updates. The auditing for existing tables is not affected.

```
sp_audit "update", "all", "default table", "on"
```

Usage

- sp_audit determines what will be audited when auditing is enabled. No actual auditing takes place until you use sp_configure to set the auditing parameter to on. Then, all auditing options that have been configured with sp_audit take effect. For more information, see sp_configure.

- If you are not the owner of the object being specified, qualify the *object_name* parameter value with the owner's name, in the following format:

  ```
  "ownername.objname"
  ```

- You cannot activate default auditing for the following options in the tempdb database:
  - delete
  - insert
  - select
  - update
  - exec_procedure
  - exec_trigger
- Table 1-6 lists the configuration parameters that control auditing.

Table 1-6: Configuration parameters that control auditing

Configuration parameter	Effect
auditing	Enables or disables auditing for the server.
audit queue size	Establishes the size of the audit queue.
current audit table	Sets the current audit table. Adaptive Server writes all audit records to that table.
suspend auditing when full	Controls the behavior of the audit process when an audit device becomes full.

All auditing configuration parameters are dynamic and take effect immediately.

Permissions

Only a System Security Officer can execute sp_audit.

See also

Documents For more information about configuring Adaptive Server for auditing, see sp_configure in the *System Administration Guide*.

System procedures sp_addauditrecord, sp_configure, sp_addaudittable

Utility commands bcp

sp_autoconnect

Description

Component Integration Services only Defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.

Syntax

sp_autoconnect *server*, {true|false}
    [, *loginame*]

Parameters

server

is the name of a server to which an automatic passthrough connection is made. *server* must be the name of a remote server already added by sp_addserver. This server cannot be the local server.

true | false

determines whether the automatic passthrough connection is enabled or disabled for *server*. true enables the automatic connection. false disables it.

loginame

specifies the name of the user for which automatic connection is required. If no *loginame* is supplied, the autoconnect status is modified for the current user.

Examples

Example 1 The current user is automatically connected to the server SYBASE the next time that user logs in. The user's connection is placed in passthrough mode:

```
sp_autoconnect SYBASE, true
```

Example 2 Disables the autoconnect feature for the user "steve":

```
sp_autoconnect SYBASE, false, steve
```

Usage

- sp_autoconnect defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.
- The System Administrator must grant connect to permission to the login prior to executing sp_autoconnect.
- Use sp_autoconnect only when Component Integration Services is installed and configured.
- Do not change the autoconnect status of the "sa" login account.
- Changing the autoconnect status does not occur immediately for users who are currently connected. They must disconnect from the local server, then reconnect before the change is made.
- Use disconnect to exit passthrough mode.

Permissions
Only a System Administrator can execute sp_autoconnect.

See also
Commands connect to...disconnect, grant

System procedures sp_addlogin, sp_addserver, sp_passthru, sp_remotesql

sp_bindcache

Description

Binds a database, table, index, text object, or image object to a data cache.

Syntax

sp_bindcache *cachename*, *dbname*
    [, [*ownername*.]*tablename*
    [, *indexname* | "text only"]]

Parameters

cachename
is the name of an active data cache.

dbname
is the name of the database to be bound to the cache or the name of the database containing the table, index, text or image object to be bound to the cache.

ownername
is the name of the table's owner. If the table is owned by "dbo", the owner name is optional.

tablename
is the name of the table to be bound to the cache, or the name of the table whose index, text object, or image object is to be bound to the cache.

indexname
is the name of the index to be bound to the cache.

text only
binds text or image objects to a cache. When this parameter is used, you cannot give an index name at the same time.

Examples

Example 1 Binds the titles table to the cache named pub_cache:

```
sp_bindcache pub_cache, pubs2, titles
```

Example 2 Binds the clustered index titles.title_id_cix to the pub_ix_cache:

```
sp_bindcache pub_ix_cache, pubs2, titles, title_id_cix
```

Example 3 Binds pubs2 to the tempdb_cache:

```
sp_bindcache tempdb_cache, pubs2
```

Example 4 Binds the pubs2 transaction log, syslogs, to the cache named logcache:

```
sp_bindcache logcache, pubs2, syslogs
```

Example 5 Binds the image chain for the au_pix table to the cache named pub_cache:

```
sp_bindcache pub_cache, pubs2, au_pix, "text only"
```

Usage

- A database or database object can be bound to only one cache. You can bind a database to one cache and bind individual tables, indexes, text objects, or image objects in the database to other caches. The database binding serves as the default binding for all objects in the database that have no other binding. The data cache hierarchy for a table or index is as follows:
    - If the object is bound to a cache, the object binding is used.
    - If the object is not bound to a cache, but the object's database is bound to a cache, the database binding is used.
    - If neither the object nor its database is bound to a cache, the default data cache is used.
- The cache and the object or database being bound to it must exist before you can execute sp_bindcache. Create a cache with sp_cacheconfig and restart Adaptive Server before binding objects to the cache.
- Cache bindings take effect immediately, and do not require a restart of the server. When you bind an object to a data cache:
    - Any pages for the object that are currently in memory are cleared.
    - When the object is used in queries, its pages are read into the bound cache.
- You can bind an index to a different cache than the table it references. If you bind a clustered index to a cache, the binding affects only the root and intermediate pages of the index. It does not affect the data pages (which are, by definition, the leaf pages of the index).
- To bind a database, you must be using the master database. To bind tables, indexes, text objects, or image objects, you must be using the database where the objects are stored.
- To bind any system tables in a database, you must be using the database and the database must be in single-user mode. Use the command:

  ```
  sp_dboption db_name, "single user", true
  ```

  For more information, see sp_dboption.
- You do not have to unbind objects or databases in order to bind them to a different cache. Issuing sp_bindcache on an object that is already bound drops the old binding and creates the new one.

- sp_bindcache needs to acquire an exclusive table lock when you are binding a table or its indexes to a cache so that no pages can be read while the binding is taking place. If a user holds locks on a table, and you issue sp_bindcache on that object, the task doing the binding sleeps until the locks are released.
- When you bind or unbind an object, all stored procedures that reference the object are recompiled the next time they are executed. When you change the binding for a database, all stored procedures that reference objects in the bound database are recompiled the next time they are executed.
- When you drop a table, index, or database, all associated cache bindings are dropped. If you re-create the table, index, or database, you must use sp_bindcache again if you want it bound to a cache.
- If a database or a database object is bound to a cache, and the cache is dropped, the cache bindings are marked invalid, but remain stored in the sysattributes system table(s). Warnings are printed in the error log when Adaptive Server is restarted. If a cache of the same name is created, the bindings become valid when Adaptive Server is restarted.
- The following procedures provide information about the bindings for their respective objects: sp_helpdb for databases, sp_help for tables, and sp_helpindex for indexes. sp_helpcache provides information about all objects bound to a particular cache.
- Use sp_spaceused to see the current size of tables and indexes, and sp_estspace to estimate the size of tables that you expect to grow. Use sp_cacheconfig to see information about cache size and status, and to configure and reconfigure caches.
- Although you can still use sp_bindcache on a system tempdb, the binding of the system tempdb is now non-dynamic. Until you restart the server:
    - The changes do not take effect
    - sp_helpcache reports a status of “P” for pending, unless you have explicitly bound the system tempdb to the default data cache, in which case the status as “V” for valid, because by default the system tempdb is already bound to the default datacache.

Restrictions

- The master database, the system tables in master, and the indexes on the system tables in master cannot be bound to a cache. You can bind non-system tables from master, and their indexes, to caches.

- You cannot bind a database or an object to a cache if:
  - Isolation level 0 reads are active on the table
  - The task doing the binding currently has a cursor open on the table
- If a cache has the type log only, you can bind a syslogs table only to that cache. Use sp_cacheconfig to see a cache's type.

Permissions

Only a System Administrator can execute sp_bindcache.

See also

System procedures sp_cacheconfig, sp_configure, sp_help, sp_helpcache, sp_helpdb, sp_helpindex, sp_poolconfig, sp_unbindcache, sp_unbindcache_all

sp_bindefault

Description

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

Syntax

sp_bindefault *defname*, *objname* [, futureonly]

Parameters

defname

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

objname

is the name of the table and column, or user-defined datatype, to which the default is to be bound. If the *objname* 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.

Existing columns of the user-defined datatype inherit the default *defname*, unless you specify futureonly.

futureonly

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

Examples

Example 1 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:

```
sp_bindefault today, "employees.startdate"
```

Example 2 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):

```
sp_bindefault def_ssn, ssn
```

Example 3 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:

```
sp_bindefault def_ssn, ssn, futureonly
```

Usage

- 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 an Adaptive Server-supplied datatype.
- You cannot bind a default to a system table.
- 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, Adaptive Server assigns the next sequential number to the IDENTITY column.
- If binding a default to a column, give the *objname* 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. To remove defaults created with create table or alter table, use alter table to replace the default with NULL.
- Existing columns of the user-defined datatype inherit the new default unless you specify futureonly. New columns of the user-defined datatype always inherit the default. Binding a default to a user-defined datatype overrides defaults bound to columns of that type; to restore column bindings, unbind and rebind the column default.
- Statements that use a default cannot be in the same batch as their sp_bindefault statement.

Permissions

Only the object owner can execute sp_bindefault.

See also

Commands create default, create table, drop default

System procedures sp_unbindefault

sp_bindexeclass

Description

Associates an execution class with a client application, login, or stored procedure.

Syntax

sp_bindexeclass "*object_name*", "*object_type*", "*scope*", "*classname*"

Parameters

object_name
is the name of the client application, login, or stored procedure to be associated with the execution class, classname.

object_type
identifies the type of object_name. Use ap for application, lg for login, or pr for stored procedure.

scope
is the name of a client application or login, or it can be NULL for ap and lg objects. It is the name of the stored procedure owner (user name) for objects. When the object with object_name interacts with the application or login, classname attributes apply for the scope you set.

classname
specifies the type of class to associate with object_name. Values are:

- EC1, EC2, or EC3
- The name of a user-defined execution class
- ANYENGINE

Examples

Example 1 This statement specifies that Transact-SQL applications will execute with EC3 attributes for any login or application process (because the value of *scope* is NULL) that invokes isql, unless the login or application is bound to a higher execution class:

```
sp_bindexeclass 'isql', 'ap', NULL, 'EC3'
```

Example 2 This statement specifies that when a login with the System Administrator role executes Transact-SQL applications, the login process executes with EC1 attributes. If you have already executed the statement in the first example, then any other login or client application that invokes isql will execute with EC3 attributes:

```
sp_bindexeclass 'sa', 'lg', 'isql', 'EC1'
```

Example 3 This statement assigns EC3 attributes to the stored procedure named my_proc owned by user kundu:

```
sp_bindexeclass 'my_proc', 'PR', 'kundu', 'EC3'
```

Usage

- sp_bindexeclass associates an execution class with a client application, login, or stored procedure. Create execution classes with sp_addexeclass.
- When scope is NULL, object_name has no scope. classname's execution attributes apply to all of its interactions. For example, if object_name is an application name, the attributes apply to any login process that invokes the application. If object_name is a login name, the attributes apply to a particular login process for any application invoked by the login process.
- When binding a stored procedure to an execution class, you must use the name of the stored procedure owner (user name) for the scope parameter. This narrows the identity of a stored procedure when there are multiple invocations of it in the same database.
- Due to precedence and scoping rules, the execution class being bound may or may not have been in effect for the object called object_name. The object automatically binds itself to another execution class, depending on other binding specifications, precedence, and scoping rules. If no other binding is applicable, the object binds to the default execution class, EC2.
- It is possible to use sp_bindexeclass to bind a RepAgent thread to an execution class using rep agent as the application without generating an error. However, because of restrictions in Adaptive Server, the priority attribute is set to medium, and the binding has no effect.
- Binding fails when you attempt to bind an active process to an engine group with no online engines.
- Adaptive Server creates a row in the sysattributes table containing the object ID and user ID in the row that stores data for the binding.
- A stored procedure must exist before it can be bound.
- Stored procedure bindings must be done in the database in which the stored procedure resides. Therefore, when binding system procedures, execute sp_bindexeclass from within the sybsystemprocs database.
- Only the "priority attribute" of the execution class is used when you bind the class to a stored procedure.
- The name of the owner of a stored procedure must be supplied as the scope parameter when you are binding a stored procedure to an execution class. This helps to uniquely identify a stored procedure when multiple stored procedures with the same name (but different owners) exist in the database.

Permissions

Only a System Administrator can execute sp_bindexeclass.

See also

System procedures sp_addexeclass, sp_showexeclass, sp_unbindexeclass

Utility isql

sp_bindmsg

Description	Binds a user message to a referential integrity constraint or check constraint.
Syntax	sp_bindmsg *constrname*, *msgid*

Parameters

constrname
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.

msgid
is the number of the user message to be bound to an integrity constraint. The message must exist in the sysusermessages table in the local database prior to calling sp_bindmsg.

Examples

```
sp_bindmsg positive_balance, 20100
```

Binds user message number 20100 to the positive_balance constraint.

Usage

- 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.
- You cannot bind a message to a unique constraint because a unique constraint does not have a constraint row in sysconstraints (a unique constraint is a unique index).
- Use the sp_addmessage procedure to insert user messages into the sysusermessages table.
- The sp_getmessage procedure retrieves message text from the sysusermessages table.
- sp_help *tablename* displays all constraint names declared on *tablename*.

Permissions

Only the object owner can execute sp_bindmsg.

See also

Commands alter table, create table

System procedures sp_addmessage, sp_getmessage, sp_unbindmsg

sp_bindrule

Description

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

Syntax

sp_bindrule *rulename*, *objname* [, 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.

objname

is the name of the table and column, or user-defined datatype, to which the rule is to be bound. If *objname* 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 you bind a rule to a user-defined datatype. It is meaningless when you bind a rule to a column.

Examples

Example 1 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:

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

Example 2 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):

```
sp_bindrule rule_ssn, ssn
```

Example 3 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:

```
sp_bindrule rule_ssn, ssn, futureonly
```

Usage

- Create a rule using the create rule statement. Then execute sp_bindrule to bind it to a column or user-defined datatype in the current database.
- Rules are enforced when an insert is attempted, not when sp_bindrule is executed. 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 an Adaptive Server-supplied datatype or to a text or an image column.
- You cannot bind a rule to a system table.
- If you are binding to a column, the *objname* 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 datatypes. Binding a rule to a column replaces a rule bound to the datatype of that column; however, binding a rule to a datatype does not replace a rule bound to a column of that user-defined datatype.
- 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.

Permissions

Only the object owner can execute sp_bindrule.

See also

Commands create rule, drop rule

System procedures sp_unbindrule

sp_cacheconfig

Description

Creates, configures, reconfigures, and drops data caches, and provides information about them.

Syntax

sp_cacheconfig [*cachename* [,"*cache_size*[P|K|M|G]"]
    [,logonly | mixed] [,strict | relaxed]]
    [, "cache_partition=[1|2|4|8|16|32|64]"]

Parameters

cachename
is the name of the data cache to be created or configured. Cache names must be unique, and can be up to 30 characters long. A cache name does not have to be a valid Adaptive Server identifier, that is, it can contain spaces and other special characters.

cache_size
is the size of the data cache to be created or, if the cache already exists, the new size of the data cache. The minimum size of a cache is 256 times the logical page size of the server. Size units can be specified with P for pages, K for kilobytes, M for megabytes, or G for gigabytes. The default is K. For megabytes and gigabytes, you can specify floating-point values. The cache size is in multiples of the logical page size.

logonly | mixed
specifies the type of cache.

strict | relaxed
specifies the cache replacement policy.

cache_partition
specifies the number of partitions to create in the cache.

Examples

Example 1 Creates the data cache pub_cache with 10MB of space. All space is in the default logical page size memory pool:

```
sp_cacheconfig pub_cache, "10M"
```

Example 2 Reports the current configuration of pub_cache and any memory pools in the cache:

```
sp_cacheconfig pub_cache
```

Example 3 Drops pub_cache at the next start of Adaptive Server:

```
sp_cacheconfig pub_cache, "0"
```

Example 4 Creates pub_log_cache and sets its type to logonly in a single step:

```
sp_cacheconfig pub_log_cache, "2000K", logonly
```

Example 5 The first command creates the cache pub_log_cache with the default type mixed. The second command changes its status to logonly. The resulting configuration is the same as that in example 4:

```
sp_cacheconfig pub_log_cache, "2000K"
sp_cacheconfig pub_log_cache, logonly
```

Example 6 Creates a cache and sets the size, type, replacement policy and number of cache partitions:

```
sp_cacheconfig 'newcache', '50M', mixed, strict, "cache_partition=2"
```

Usage

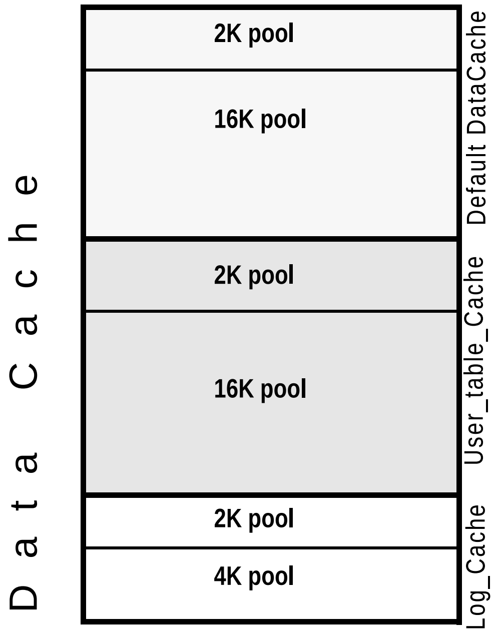
- The minimum cache size is 256 times the logical page size. For example, a 4K server would have a minimum cache size of 1024K.
- If Adaptive Server is unable to allocate all the memory requested while you are creating a new cache or adding memory to an existing cache, it allocates all the available memory. However, this additional memory is allocated at the next reboot of Adaptive Server.
- If there are objects bound to cache (including the default cache), you cannot delete the cache until you unbind the objects.
- Some of the actions you perform with sp_cacheconfig are dynamic (do not require a reboot of Adaptive Server) and some are static (require a reboot). Table 1-7 describes which are dynamic and which are static:

Table 1-7: Dynamic and static sp_cacheconfig actions

Dynamic sp_cacheconfig actions	Static sp_cacheconfig actions
Adding a new cache	Changing the number of cache partitions
Adding memory to an existing cache	Reducing a cache size
Deleting a cache	Changing the replacement policy
Changing a cache type	

- When you first create a data cache:
    - All space is allocated to the logical page size memory pool.
    - The default type is mixed.
- Figure 1-1 shows a data cache for a 2K server with two user-defined data caches configured and the following pools:
    - The default data cache with a 2K pool and a 16K pool
    - A user cache with a 2K pool and a 16K pool
    - A log cache with a 2K pool and a 4K pool

Figure 1-1: Data cache with default and user-defined caches



- The default data cache must always have the type default, and no other cache can have the type default.
- The Adaptive Server housekeeper task does not do any buffer washing in caches with a type of logonly or in caches with a relaxed LRU replacement policy.
- The following commands perform only 2K I/O: disk init, some dbcc commands, and drop table. The dbcc checkdb and dbcc checktable commands can perform large I/O for tables, but perform 2K I/O on indexes. Table 1-8 shows cache usage, depending on the binding of the database or object.

Table 1-8: Cache usage for Transact-SQL commands

Command	Database bound	Table or index is bound	Database or object not bound
create index	Bound cache	N/A	Default data cache
disk init	N/A	N/A	Default data cache
dbcc checkdb	Bound cache	N/A	Default data cache

Command	Database bound	Table or index is bound	Database or object not bound
dbcc checktable, indexalloc, tablealloc	Bound cache	Bound cache	Default data cache
drop table	Bound cache	Bound cache	Default data cache

- Recovery uses only the logical page size pool of the default data cache. All pages for all transactions that must be rolled back or rolled forward are read into and changed in this pool. Be sure that your default logical page size pool is large enough for these transactions.
- When you use sp_cacheconfig with no parameters, it reports information about all of the caches on the server. If you specify only a cache name, it reports information about only the specified cache. If you use a fragment of a cache name, it reports information for all names matching "%*fragment*%".

  All reports include a block of information that reports information about caches, and a separate block of data for each cache that provides information about the pools within the cache.

  The output below, from a server using 2K, shows the configuration for:

  - The default data cache with two pools: a 2K pool and a 16K pool. The default data cache has 2 partitions.
  - pubs_cache with two pools: 2K and 16K
  - pubs_log, with the type set to logonly and cache replacement policy set to relaxed, with a 2K pool and a 4K pool

```
Cache Name               Status    Type     Config Value  Run Value
 ----------------------- --------- -------- ------------ --------
default data cache        Active   Default      0.00 Mb   26.09 Mb
pubs_cache               Active    Mixed       10.00 Mb  10.00 Mb
pubs_log                 Active    Log Only     2.40 Mb    2.40 M
                                              ------------ --------
                                   Total       12.40 Mb   38.49 Mb
===================================================================
Cache: default data cache,   Status: Active,   Type: Default
      Config Size: 0.00 Mb,   Run Size: 26.09 Mb
      Config Replacement: strict LRU,   Run Replacement: strict LRU
      Config Partition:           2,   Run Partition:           2
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb   3704 Kb      0.00 Mb     18.09 Mb     10
    16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
===================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
      Config Size: 10.00 Mb,   Run Size: 10.00 Mb
      Config Replacement: strict LRU,   Run Replacement: strict LRU
      Config Partition:           1,   Run Partition:           1
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb   1228 Kb      0.00 Mb      6.00 Mb     10
    16 Kb    816 Kb      4.00 Mb      4.00 Mb     10
===================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
      Config Size: 2.40 Mb,   Run Size: 2.40 Mb
      Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
      Config Partition:           1,   Run Partition:           1
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb    206 Kb      0.00 Mb      1.01 Mb     10
    16 Kb    272 Kb      1.40 Mb      1.39 Mb     10
```

Table 1-9 lists the meaning of the columns in the output:

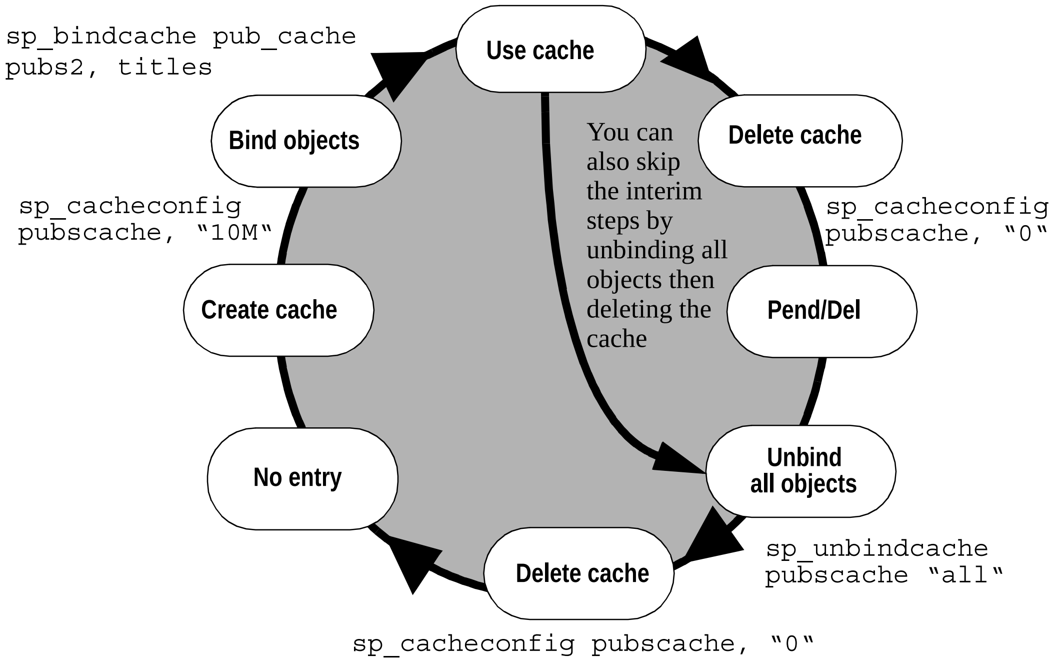
Table 1-9: sp_cacheconfig output

Column	**Meaning**
Cache Name	The name of the cache.
Status	One of the following: • "Active" • "Pend/Act" • "Pend/Del" These are explained following this table.
Type	"Mixed" or "Log Only" for user-defined caches, "Default" for the default data cache.
I/O Size	The size of I/O for a memory pool. This column is blank on the line that shows that cache configuration.
Wash Size	The size of the wash area for the pool. As pages enter the wash area of the cache, they are written to disk. This column is blank on the line that shows the cache configuration.
Config Value or Config Size	The size that the cache or pool. If the value is 0, the size has not been explicitly configured, and a default value will be used.
Run Value or Run Size	The size of the cache or pool now in use on Adaptive Server.
Config/ Run Replacement	The cache policy (strict or relaxed) that will be used for the cache after the next restart, and the current replacement policy. These will be different only if the policy has been changed since the last reboot.

Column	Meaning
Config/Run Partition	The number of cache partitions that will be used for the cache, and the current number of partitions. These will be different if sp_cacheconfig has been used to change the number of partitions since the last reboot.
APF Percent	The percentage of buffers in the pool that can hold buffers that have been fetched by asynchronous prefetch, but have not been used.
Total	The total size of data cache, if the report covers all caches, or the current size of the particular cache, if you specify a cache name.

The status “Pend” is short for pending. It always occurs in combination with either “Act” for Active or “Del” for Delete. It indicates that a configuration action has taken place, but that the server must be restarted in order for the changes to take effect.

Figure 1-2: Effects of restarts and sp_cacheconfig on cache status


sp_bindcache pub_cache pubs2, titles
Use cache
Delete cache
Bind objects
You can also skip the interim steps by unbinding all objects then deleting the cache
sp_cacheconfig pubscache, “10M”
sp_cacheconfig pubscache, “0”
Create cache
Pend/Del
No entry
Unbind all objects
sp_unbindcache pubscache “all”
Delete cache
sp_cacheconfig pubscache, “0”

- You can also configure caches and pools by editing the configuration file. For more information, see the *System Administration Guide*.

Data cache memory

- When Adaptive Server is first installed, all data cache memory is assigned to the logical page size pool of the cache named default data cache. The default data cache is used by all objects that are not explicitly bound to a data cache with sp_bindcache or whose databases are not bound to a cache.
- When you create data caches, the memory allocation is validated against max memory. Memory for caches is allocated out of the memory allocated to Adaptive Server with the total logical_memory configuration parameter. To increase the amount of space available for caches, increase total logical memory, or decrease other configuration settings that use memory. If the sum of total logical memory and additional memory requested is greater than max memory, then Adaptive Server issues and error and does not perform the changes.

  The default cache is used for all objects, including system tables, that are not bound to another cache, and is the only cache used during recovery. For more information, see the *System Administration Guide*.
- A data cache requires a small percentage of overhead for structures that manage the cache. All cache overhead is taken from the default data cache. To see the amount of overhead required for a specific size of cache, use sp_helpcache, giving the size:

  ```
  sp_helpcache "200M"
  10.38Mb of overhead memory will be needed to manage
  a cache of size 200M
  ```

  This is only an estimate of the overhead. The actual overhead may be larger because of runtime issues.

Changing existing caches

- To change the size of an existing cache, specify the cache’s name and the new size.
  - If you increase the size of an existing cache, all of the added space is placed in the smallest pool.
  - To reduce the size of an existing cache, all of the space must be available in the logical page size pool. You may need to use sp_poolconfig to move space from other pools to this pool.
- If you have a database or any nonlog objects bound to a cache, you cannot change its type to logonly.

Using cache partitions

- Cache partitions can be used to reduce cache spinlock contention without needing to create separate caches and bind database objects to them. For more information on monitoring cache spinlock contention, see the *Performance and Tuning Guide*.
- You can set the default number of cache partitions for all caches with the configuration parameter global cache partition number. See the *System Administration Guide*.

Dropping caches

- To drop or delete a data cache, change its size to 0, as shown in example 3. When you set a cache's size to 0, the cache is marked for deletion. The cache remains active, and all objects that are bound to that cache continue to use it.

  You cannot drop the default data cache.
- If you delete a data cache, and there are objects bound to the cache, the cache is left as-is in memory and Adaptive Server issues the following message:

```
Cache (nmc3) not deleted dynamically. Objects are bound to the cache. Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

  The entry corresponding to the cache in the configuration file is deleted, as well as the entries corresponding to the cache in sysconfigures, and the cache is deleted the next time Adaptive Server is restarted.
- You cannot run sp_cacheconfig within a transaction.

Permissions

Only a System Administrator can execute sp_cacheconfig to change cache configurations. Any user can execute sp_cacheconfig to view cache configurations.

See also

System procedures sp_bindcache, sp_helpcache, sp_poolconfig, sp_unbindcache, sp_unbindcache_all

sp_cachestrategy

Description

Enables or disables prefetching (large I/O) and MRU cache replacement strategy for a table, index, text object, or image object.

Syntax

sp_cachestrategy *dbname*, [*ownername*.]*tablename*
  [, *indexname* | "text only" | "table only"
  [, { prefetch | mru }, { "on" | "off"}]]

Parameters

dbname
is the name of the database where the object is stored.

ownername
is the name of the table's owner. If the table is owned by "dbo", the owner name is optional.

tablename
is the name of the table.

indexname
is the name of the index on the table.

text only
changes the cache strategy for a text or image object.

table only
changes the cache strategy for a table.

prefetch | mru
is prefetch or mru, and specifies which setting to change.

on | off
specifies the setting, "on" or "off", enclosed in quotes.

Examples

Example 1 Displays information about cache strategies for the titles table:

```
sp_cachestrategy pubs2, titles

object name        index name    large IO MRU
------------------ ------------- -------- --------
dbo.titles           titleidind     ON         ON
```

Example 2 Displays information about cache strategies for the titleind index:

```
sp_cachestrategy pubs2, titles, titleind
```

Example 3 Disables prefetch on the titleind index of the titles table:

```
sp_cachestrategy pubs2, titles, titleind, prefetch, "off"
```

Example 4 Reenables MRU replacement strategy on the authors table:

```
sp_cachestrategy pubs2, authors, "table only", mru, "on"
```

Example 5 Reenables prefetching on the text pages of the blurbs table:

```
sp_cachestrategy pubs2, blurbs, "text only", prefetch, "on"
```

Usage

- If memory pools for large I/O are configured for the cache used by a table or an index, the optimizer can choose to prefetch data or index pages by performing large I/Os of up to eight data pages at a time. This prefetch strategy can be used on the data pages of a table or on the leaf-level pages of a nonclustered index. By default, prefetching is enabled for all tables, indexes, and text or image objects. Setting the prefetch option to "off" disables prefetch for the specified object.
- The optimizer can choose to use **MRU replacement strategy** to fetch and discard buffers in cache for table scans and index scans for I/O of any size. By default, this strategy is enabled for all objects. Setting mru to "off"disables this strategy. If you turn mru off for an object, all pages are read into the MRU/LRU chain in cache, and they remain in the cache until they are flushed by additional I/O. For more information on cache strategies, see the *Performance and Tuning Guide*.
- You can change the cache strategy only for objects in the current database.
- When you use sp_cachestrategy without specifying the strategy and setting, it reports the current settings for the object, as shown in Example 1.
- To see the size, status and I/O size of all data caches on the server, use sp_cacheconfig.
- Setting prefetch "on" has no effect on tables or indexes that are read into a cache that allows only 2K I/O. The mru strategy can be used in all caches, regardless of available I/O size.

Overrides

- If prefetching is turned on for a table or an index, you can override the prefetching for a session with set prefetch "off". If prefetching is turned off for an object, you cannot override that setting.
- The prefetch, lru, and mru options to the select, delete and update commands suggest the I/O size and cache strategy for individual statements. If prefetching or MRU strategy is enabled for a table or an index, you can override it for a query by specifying I/O the size of the logical page size for prefetch, and by specifying lru strategy. For example, the following command forces LRU strategy, logical page size I/O, and a table scan of the titles table:

```
select avg(advance)
from titles (index titles prefetch 2 lru)
```

If you request a prefetch size, and the object's cache is not configured for I/O of the requested size, the optimizer chooses the best available I/O size.

- If prefetching is enabled for an object with sp_cachestrategy, using a prefetch specification of the logical page size in a select, update or delete command overrides an earlier set prefetch "on" statement. Specifying a larger I/O size in a select, update or delete command does not override a set prefetch "off" command.

Permissions

Only a System Administrator or the object owner can execute sp_cachestrategy.

See also

Commands delete, select, set, update

Stored procedures sp_cacheconfig, sp_poolconfig

sp_changedbowner

Description

Changes the owner of a user database.

Syntax

sp_changedbowner *loginame* [, true]

Parameters

loginame
is the login name of the new owner of the current database.

true
transfers aliases and their permissions to the new database owner. Values are “true” and “TRUE”.

Examples

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

```
sp_changedbowner albert
```

Usage

- 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 *loginame* changes the database ownership to *loginame* and drops aliases of users who could act as the old “dbo.”
- After executing sp_changedbowner, the new owner is known as the Database Owner inside the database.
- sp_changedbowner cannot transfer ownership of the system databases.
- The new owner must already have a login name in Adaptive 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.

Permissions

Only a System Administrator can execute sp_changedbowner.

See also

Commands create database

System procedures sp_addlogin, sp_dropalias, sp_dropuser, sp_helpdb

sp_changegroup

Description

Changes a user's group.

Syntax

sp_changegroup *grpname*, *username*

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 keyword.

username
is the name of the user to be added to the group. The user must already exist in the current database.

Examples

Example 1 The user "albert" is now a member of the "fort_mudge" group. It doesn't matter what group "albert" belonged to before:

```
sp_changegroup fort_mudge, albert
```

Example 2 Removes "albert" from the group he belonged to without making him a member of a new group (all users are always members of "public"):

```
sp_changegroup "public", albert
```

Usage

- 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 is 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 that user a member of a new group, use sp_changegroup to change the user's group to "public", as shown above in Example 2.
- 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.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute sp_changegroup.

See also

Commands grant, revoke

System procedures sp_addgroup, sp_adduser, sp_dropgroup, sp_helpgroup

sp_checknames

Description

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

Syntax

sp_checknames

Parameters

None.

Examples

```
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
```

Usage

- 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 SQL Server of release 4.0.x or 4.2.x, and after using a default character set that was not 7-bit ASCII.
- Follow the instructions in the sp_checknames report to correct all non-ASCII names.

Permissions

Any user can execute sp_checknames.

See also

Commands update

System procedures sp_password, sp_rename, sp_renamedb

sp_checkreswords

Description

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 [*user_name_param*]

Parameters

user_name_param

is the name of a user in the current database. If you supply *user_name_param*, sp_checkreswords checks only for objects that are owned by the specified user.

Examples

Example 1 Shows the results if sp_checkreswords is executed in the master database:

```
1> /* executed in the master database */
2> sp_checkreswords
```

```
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
```

Example 2 Shows the results if sp_checkreswords is executed in the user database user_db:

```
1> /* executed in the user database, user_db */
2> sp_checkreswords
```

```
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.
```

Usage

- 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 as identifiers, unless you are using delimited identifiers. Reserved words are pieces of SQL syntax, and they have special meaning when you use them as part of a command. For example, in pre-release 10.0 SQL Server, you could have a table called work, and select data from it with this query:

  ```
  select * from work
  ```

  work was a new reserved word in SQL Server release 10.0, part of the command commit work. Issuing the same select statement in release 10.0 or later causes a syntax error. sp_checkreswords finds identifiers that would cause these problems.
- sp_checkreswords also finds reserved words, used as identifiers, that were created using the set quoted_identifier option.
- Use sp_checkreswords before or immediately after upgrading to a new release of Adaptive Server. For information on installing and running this procedure before performing the upgrade, see the installation documentation for your platform.

  Run sp_checkreswords in the master database and in each user database. Also run it in model and sybsystemprocs, if you have added users or objects to those databases.
- The return status indicates the number of items found.
- If you supply a user name, sp_checkreswords checks for all of the objects that can be owned by a user tables, indexes, views, procedures, triggers, rules, defaults, and user-defined datatypes. It reports all identifiers that are reserved words.

- If your current database is not the master database, 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 and sybsystemprocs if you have added objects, users, or user-defined datatypes.
- 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, and languages.

Handling reported instances of reserved words

- If sp_checkreswords reports that reserved words are used as identifiers, you have two options:
  - Use sp_rename, sp_renamedb, or update the system tables to change the name of the identifier.
  - Use set quoted_identifier on 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 set quoted_identifier on, and quote all identifiers. All users will be able to run the procedures, without having to use set quoted_identifier on for their session. You can use set quoted_identifier on, 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"
    ```

    The syntax for the set command is:

    ```
    set quoted_identifier on
    ```

- If you do not either 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'.
```

Note The quoted identifier option is a SQL92 option and may not be supported by many client products that support other Adaptive 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 the tools you will use to access Adaptive Server. Use set quoted_identifier on, create a table with a reserved word for a name and reserved words for 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 been changed may work after the name change, but will stop working when the query plan is recompiled. Recompilation takes place for many reasons, without notification to the user. To avoid unsuspected loss of functionality, change the names of objects in procedures, triggers, and views immediately after you change the object name.
- Whether you 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 identifiers, you must add the set quoted_identifier option and quotation marks.
- If you do not have the text of your procedures, triggers, views, rules, and defaults saved in operating system files, you can use defncopy to copy the definitions from the server to files. See defncopy in the *Utility Guide*.

Changing identifiers

- If you change the names of the items reported by sp_checkreswords, you must change the names in all 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 the 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, 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 to rename databases.
- Table 1-10 shows the types of identifiers that you can change with sp_rename and lists other changes that may have to be made on the server and in your application programs.

Table 1-10: sp_rename and changing identifiers

Identifier	Remember To
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.
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.

Identifier	Remember To
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-create them 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.

The following command changes the name of the view isolation to isolated:

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

The following command changes the name of a column in the renamed view isolated:

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

- Use sp_depends to get a list of all views, procedures, and triggers that reference a view, procedure, or table that will 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 re-create any procedures, triggers, and views that explicitly reference the database name. For more information, see sp_renamedb.

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, then add or create it again. If you cannot do that, use the following command to allow direct updates to system tables:

```
sp_configure "allow updates to system tables", 1
```

Only a System Security Officer can set the allow updates to system tables configuration parameter.

Errors during direct updates to system tables can create severe problems in Adaptive Server. To determine whether you can drop the objects or user, then re-create them, see Table 1-11.

Table 1-13 on page 114 shows possible dependencies on this set of identifiers. See 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.

Table 1-11: Alternatives to direct system tables updates when changing identifiers

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). Finally, 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, you will not need to use its name in a create database command, so 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 have reserved words as identifiers only if a System Administrator has created alternative languages with sp_addlanguage. Drop the language with sp_droplanguage, and add it with a new name.

Warning! Direct updates to system tables can be very dangerous. You can make mistakes that make it impossible for Adaptive Server to run or make it impossible to access objects in your databases. Undertake this effort when you are calm and collected, and when little or no production activity is taking place on the server. If possible, use the alternative methods described Table 1-11.

- The following example shows a "safe" procedure for updating a user name, with all data modification preceded by a begin transaction command. The System Security Officer executes the following command:

```
sp_configure "allow updates to system tables", 1
```

Then you can execute the following:

```
begin transaction
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.

When you are finished, the System Security Officer turns off the allow updates to system tables configuration parameter with this command:

```
sp_configure "allow updates to system tables", 0
```

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.

Table 1-12 shows the system tables and columns that you should update 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 databases. Be certain you are using the correct database before you attempt the update. You can check for the current database name with this command:

```
select db_name()
```

Table 1-12: System table columns to update when changing identifiers

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 1-13 shows other changes that may have to be made on the server and in your application programs:

Table 1-13: Considerations when changing identifiers

Identifier	Remember to
Login name	Change the user name in each database where this person is a user.
User name	Drop, edit, and re-create 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. 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	Change the name on the remote server. If the name that sp_checkreswords reports is the name of the local server, you must restart 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	Change the name on the remote server.
Segment name	Drop and re-create 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.

Using delimited identifiers

- You can use delimited identifiers for table names, column names, and view names. You cannot use delimited identifiers for other object names.

- If you choose to use delimited identifiers, use set quoted_identifier on, and drop and re-create all the procedures, triggers, and views that use the identifier. Edit the text for those objects, enclosing the reserved words in double quotes and enclosing all character strings in single quotes.

  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 pre-release 10.0 query, which encloses 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 or later version of query, using
  ** the quoted identifier 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 used by the application while the quoted identifier option is turned on must be enclosed in single quotes. Otherwise, Adaptive 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%"
  ```

  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 to system tables option also works this way.) 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.

Permissions

Only a System Administrator can execute sp_checkreswords.

See also

Commands set

System procedures sp_configure, sp_depends, sp_rename, sp_renamedb

Utilities defncopy

sp_checksource

Description

Checks for the existence of the **source text** of the **compiled object**.

Syntax

sp_checksource [objname [, tabname [, username]]]

Parameters

objname
is the compiled object to be checked for the existence of its source text.

tabname
is the name of the table or view to be checked for the existence of all check constraints, defaults, and triggers defined on it.

username
is the name of the user who owns the compiled objects to be checked for the existence of the source text.

Examples

Example 1 Checks for the existence of the source text of all compiled objects in the current database:

```
sp_checksource
```

Example 2 Checks for the existence of the source text of the view named titleview:

```
sp_checksource titleview
```

Example 3 Checks for the existence of the source text of the view named titls_vu that is owned by Mary:

```
sp_checksource title_vu, @username = Mary
```

Example 4 Checks for the existence of the source text of the custom stored procedure list_phone_proc:

```
sp_checksource list_phone_proc
```

Example 5 Checks for the existence of the source text of all the check constraints, triggers, and declarative defaults defined on the table named my_tab:

```
sp_checksource @tabname = "my_tab"
```

Example 6 Checks for the existence of the source text of the view my_vu and all check constraints, triggers, and defaults defined on the table my_tab:

```
sp_checksource @objname = "my_vu", @tabname = "my_tab"
```

Example 7 Checks for the existence of the source text of all compiled objects owned by Tom:

```
sp_checksource @username = "Tom"
```

Usage

- sp_checksource checks for the existence of the source text of the specified compiled object. If the source text exists for the specified object, sp_checksource returns 0. If the source text does not exist for the specified object, sp_checksource returns 1.
- If you do not provide any parameters, sp_checksource checks the existence of the source text for all compiled objects in the current database.
- To use sp_checksource with no parameters, you must be the Database Owner or System Administrator.

Permissions

Only a Database Owner or System Administrator can execute sp_checksource to check for the existence of the source text of compiled objects that are owned by another user. Any user can execute sp_checksource to check for the existence of the source text for his or her own compiled objects.

See also

System procedures sp_hidetext

sp_chgattribute

Description

Changes the max_rows_per_page, fillfactor, reservepagegap, or exp_row_size value for future space allocations of a table or an index; sets the concurrency_opt_threshold for a table. Provides the user interface for optimistic index locking, which acquires an exclusive table lock on a specified table.

Syntax

```
sp_chgattribute objname, {"max_rows_per_page" | "fillfactor" |
        "reservepagegap" | "exp_row_size"
        concurrency_opt_threshold | “optimistic_index_lock”}, value,
optvalue
```

```
sp_chgattribute "table_name", {"identity_gap", set_number |
        “dealloc_first_txtpg”,1}
```

Parameters

objname
is the name of the table or index for which you want to change attributes.

max_rows_per_page
specifies the row size. Use this option for tables with variable-length columns.

fillfactor
specifies how full Adaptive Server will make each page when it is re-creating an index or copying table pages as a result of a reorg rebuild command or an alter table command to change the locking scheme. The fillfactor percentage is relevant only at the time the index is rebuilt. Valid values are 0–100.

reservepagegap
specifies the ratio of filled pages to empty pages that are to be left during extent I/O allocation operations. For each specified *num_pages*, an empty page is left for future expansion of the table. Valid values are 0–255. The default value is 0.

exp_row_size
reserves a specified amount of space for the rows in data-only locked tables. Use this option to reduce the number of rows being forwarded, which can be expensive during updates. Valid values are 0, 1, and any value between the minimum and maximum row length for the table. 0 means a server-wide setting is applied, and 1 means to fully pack the rows on the data pages.

concurrency_opt_threshold
: specifies the table size, in pages, at which access to a data-only-locked table should begin optimizing for reducing I/O, rather than for concurrency. If the table is smaller than the number of pages specified by concurrency_opt_threshold, the query is optimized for concurrency by always using available indexes; if the table is larger than the number of pages specified by concurrency_opt_threshold, the query is optimized for I/O instead. Valid values are -1 to 32767. Setting the value to 0 disables concurrency optimization. Use -1 to enforce concurrency optimization for tables larger than 32767 pages. The default is 15 pages.

optimistic_index_lock
: is the name of the option that sets an exclusive table lock on the table you specify.

value
: set 1 to turn optimistic index locking on, or 0 to turn it off.

optvalue
: is the new value. Valid values and default values depend on which parameter is specified.

table_name
: is the name of the table for which you want to change the identity gap. Used also to deallocate the text and image pages on a table to free up space.

identity_gap
: indicates that you want to change the identity gap.

set_number
: is the new size of the identity gap.

Examples

Example 1 Sets the max_rows_per_page to 1 for the authors table for all future space allocations:

```
sp_chgattribute authors, "max_rows_per_page", 1
```

Example 2 Sets the max_rows_per_page to 4 for the titleidind index for all future space allocations:

```
sp_chgattribute "titles.titleidind", "max_rows_per_page", 4
```

Example 3 Specifies a fillfactor of 90 percent for pages in title_ix:

```
sp_chgattribute "titles.title_ix", "fillfactor", 90
```

Example 4 Sets the exp_row_size to 120 for the authors table for all future space allocations:

```
sp_chgattribute "authors", "exp_row_size", 120
```

Example 5 Sets the reservepagegap to 16 for the titleidind index for all future space allocations:

```
sp_chgattribute "titles.titleidind", "reservepagegap", 16
```

Example 6 Turns off concurrency optimization for the titles table:

```
sp_chgattribute "titles", "concurrency_opt_threshold", 0
```

Example 7 Sets the identity gap for mytable to 20:

```
sp_chgattribute "mytable", "identity_gap", 20
```

Example 8 Changes mytable to use the identity burning set factor setting instead of the identity_gap setting:

```
sp_chgattribute "mytable", "identity_gap", 0
```

Sets the value of sp_chgattribute to 1, turning the optimistic index locking feature on.

```
sp_chgattribute "mytable", "optimistic_index_lock", 1
```

Sets the value of sp_chgattribute to 0, turning the optimistic index locking feature off.

```
sp_chgattribute "mytable", "optimistic_index_lock", 0
```

Example 9 Switches the deallocation for text and image space on.

```
sp_chgattribute "mytable", "deallocate_first_txtpg", 1
```

To switch the feature off:

```
sp_chgattribute "mytable", "deallocate_first_txtpg", 0
```

Usage

- sp_chgattribute changes the max_rows_per_page, fillfactor, reservepagegap, or exp_row_size value for future space allocations or data modifications of the table or index. It does not affect the space allocations of existing data pages. You can change these values for an object only in the current database.
- Use sp_help to see the stored spance management values for a table. Use sp_helpindex to see the stored space management values for an index.
- Setting max_rows_per_page to 0 tells Adaptive Server to fill the data or index pages and not to limit the number of rows (this is the default behavior of Adaptive Server if max_rows_per_page is not set).
- Low values for *optvalue* may cause page splits. Page splits occur when new data or index rows need to be added to a page, and there is not enough room for the new row. Usually, the data on the existing page is split fairly evenly between the newly allocated page and the existing page.

- To approximate the maximum value for a nonclustered index, subtract 32 from the page size and divide the resulting number by the index key size. The following statement calculates the maximum value of max_rows_per_page for the nonclustered index titleind:

  ```
  select
      (select @@pagesize - 32) / minlen
      from sysindexes where name = "titleind"
  -----------
          288
  ```

  If you specify too high a value for *optvalue*, Adaptive Server returns an error message specifying the highest value allowed.

- If you specify an incorrect value for max_rows_per_page, fillfactor, reservepagegap, or exp_row_size, sp_chgattribute returns an error message specifying the valid values.
- For more information on max_rows_per_page, fillfactor, reservepagegap, exp_row_size, and concurrency_opt_threshold, see the Performance and Tuning Guide.
- For more information about identity gaps, see the section “Managing Identity Gaps in Tables” in Chapter 7, “Creating Databases and Tables” in the *Transact-SQL User’s Guide*.
- You cannot run this stored procedure from within a transaction.
- Only a user with sa_role privileges can execute this stored procedure.
- You cannot set the optimistic index locking option for tables with datapages or datarow locking schemes.
- You cannot set the optimistic index locking option for tables in system databases, such as master or tempdb. You can set it only on user-defined tables.
- If you do not acquire a lock on the specified table, sp_chgattribute fails.
- The default property of the optimistic index locking option is off.

Permissions

Only the object owner can execute sp_chgattribute.

See also

Commands alter table, create index, create table

System procedures sp_helpindex

sp_clearpsexe

Description

Clears the execution attributes of an Adaptive Server session that was set by sp_setpsexe.

Syntax

sp_clearpsexe *spid*, *exeattr*

Parameters

spid
is the process ID of the session for which execution attributes are to be cleared.

exeattr
identifies the execution attributes to be cleared. Values for exeattr are "priority" and "enginegroup".

Examples

Drops the engine group entry for process 12.

```
sp_clearpsexe 12, 'enginegroup'
```

Usage

- If the execution attributes are not cleared during the lifetime of the session, they are cleared when the session exits or terminates abnormally.
- sp_clearpsexe fails if there are no online engines in the associated engine group.
- When you drop an engine group entry, the session executes on an engine group determined by a class definition or by the default class.
- Use sp_who to list process IDs (spids).

Permissions

Only a System Administrator can execute sp_clearpsexe to clear priority attributes for all users. Any user can execute sp_clearpsexe to clear the priority attributes of tasks owned by that user.

See also

Documents sp_clearpsexe clears the execution attributes of the session that was set by sp_setpsexe. For more information, see the *Performance and Tuning Guide*.

System procedures sp_addexeclass, sp_bindexeclass, sp_dropexeclass, sp_showexeclass, sp_unbindexeclass

sp_clearstats

Description

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 [*loginame*]

Parameters

loginame
is the user's login name.

Examples

Example 1 Initiates a new accounting period for all users.

```
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.
```

Example 2 Initiates a new accounting period for the user "kathy."

```
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.
```

Usage

- sp_clearstats creates an accounting period and should be run only at the end of a period.
- Because sp_clearstats clears out the accounting statistics, you must record the statistics *before* running the procedure.
- sp_clearstats updates the syslogins field accdate and clears the syslogins fields totcpu and totio.

Permissions

Only a System Administrator can execute sp_clearstats.

See also **System procedures** sp_reportstats

sp_client_addr

Description
Displays the IP (Internet Protocol) address of every Adaptive Server task with an attached client application, including the spid and the client host name.

Syntax
sp_client addr[“*spid*”]

Parameters
spid
specifies one task for which you require an IP address.

Examples
Example 1 Lists IP addresses for all tasks:

```
sp_client_addr

---------
spid   hostname  ipaddr
-----------------------------
11     FRED      162.66.131.36
21     BARNEY    162.66.100.233
22     WILMA     162.66.100.206
23     BETTY     162.66.100.119
24     PEBBLES   162.66.100.125
25     BAMBAM    162.66.100.124
(6 rows affected)
(return status = 0)
```

Example 2 Shows IP addresses for spid 21:

```
sp_client_addr 21

----------
spid   hostname  ipaddr
-----------------------------
21     BARNEY    162.66.100.233
(1 row affected)
(return status = 0)
```

Example 3 Shows the result when a client application is not connected via IP:

```
sp_client_addr 11

----------
spid   hostname  ipaddr
-----------------------------
11     FRED      0.0.0.0
(1 row affected)
(return status = 0)
```

Example 4 Shows the result of a task with no attached client; for example, Housekeeper:

```
sp_client_addr 9

----------
spid   hostname  ipaddr
-------------------------------
9                NULL
(1 row affected)
(return status = 0)
```

Example 5 Shows the result when an incorrect spid is specified:

```
sp_client_addr 99

----------
Msg 18934, Level 16, State 1:
Procedure "sp_client_addr", Line 32:
spid not found
(return status = 1)
```

Usage

- If the client application is not attached by IP, the address appears as 0.0.0.0. Adaptive Server does not support display of addresses of protocols other than IP.
- If a task has no attached client (Housekeeper, for instance), the IP address appears as “NULL”. Tasks with no attached client are not listed when you use sp_client_addr with no parameter.

Permissions

Any user can execute sp_client_addr.

See also

System procedures sp_who

sp_cmp_all_qplans

Description

Compares all abstract plans in two abstract plan groups.

Syntax

sp_cmp_all_qplans *group1*, *group2* [, *mode*]

Parameters

group1, *group2*
are the names of the 2 abstract plan groups.

mode
is the display option, one of: counts, brief, same, diff, first, second, offending and full. The default mode is counts.

Examples

Example 1 Generates a default report on 2 abstract plan groups:

```
sp_cmp_all_qplans dev_plans, prod_plans
```

```
If the two query plans groups are large, this might take some time.
Query plans that are the same
 count
 -----------
          49
Different query plans that have the same association key
 count
 -----------
           1
Query plans present only in group ’dev_plans’ :
 count
 -----------
           1
Query plans present only in group ’prod_plans’ :
 count
 -----------
           0
```

Example 2 Generates a report using the brief mode:

```
sp_cmp_all_qplans dev_plans, prod_plans, brief
```

Usage

- Use sp_cmp_all_qplans to check for differences in abstract plans in two groups of plans.
- sp_cmp_all_qplans matches pairs of plans where the plans in each group have the same user ID and query text. The plans are classified as follows:
  - Plans that are the same
  - Plans that have the same association key in both groups, but have different abstract plans. The association key is the group ID, user ID and query text.

- Plans that exist in one group, but do not exist in the other group

Table 1-14 shows the report modes and what type of information is reported for each mode.

Table 1-14: Report modes for sp_cmp_all_qplans

Mode	Reported information
counts	The counts of: plans that are the same, plans that have the same association key, but different groups, and plans that exist in one group, but not the other. This is the default report mode.
brief	The information provided by counts, plus the IDs of the abstract plans in each group where the plans are different, but the association key is the same, and the IDs of plans that are in one group, but not in the other.
same	All counts, plus the IDs, queries, and plans for all abstract plans where the queries and plans match.
diff	All counts, plus the IDs, queries, and plans for all abstract plans where the queries and plans are different.
first	All counts, plus the IDs, queries, and plans for all abstract plans that are in the first plan group, but not in the second plan group.
second	All counts, plus the IDs, queries, and plans for all abstract plans that are in the second plan group, but not in the first plan group.
offending	All counts, plus the IDs, queries, and plans for all abstract plans that have different association keys or that do not exist in both groups. This is the combination of the diff, first and second modes
full	All counts, plus the IDs, queries, and plans for all abstract plans. This is the combination of same and offending modes.

- To compare two individual abstract plans, use sp_cmp_qplans. To see the names of abstract plan groups, use sp_help_qpgroup.
- When a System Administrator or Database Owner runs sp_cmp_all_qplans, it reports on all plans in the two groups. When another user executes sp_cmp_all_qplans, it reports only on plans that have the user's ID.

Permissions

Any user can execute sp_cmp_all_qplans.

See also

System procedures sp_cmp_qplans, sp_help_qpgroup

sp_cmp_qplans

Description

Compares two abstract plans.

Syntax

sp_cmp_qplans *id1*, *id2*

Parameters

id1, *id2*
are the IDs of two abstract plans.

Examples

Example 1 Compares abstract plan 411252620 to 1383780087:

```
sp_cmp_qplans 411252620, 1383780087

The queries are the same.
The query plans are the same.
```

Example 2 Compares abstract plan 2091258605 to 647777465:

```
sp_cmp_qplans 2091258605, 647777465

The queries are the same.
The query plans are different.
```

Usage

- sp_cmp_qplans compares the queries, abstract plans, and hash keys of two abstract plans, and reports whether the queries are the same, and whether the plans are the same. It prints one of these messages for the query:
  - The queries are the same.
  - The queries are different.
  - The queries are different but have the same hash key.

  It prints one of these messages for the abstract plan:
  - The query plans are the same.
  - The query plans are different.
- sp_cmp_qplans also prints a return status showing the results of the comparison. The status values 1, 2 and 10 are additive. The status values are show in Table 1-15

Table 1-15: Return status values for sp_cmp_qplans

Return value	Meaning
0	The query text and abstract plans are the same.
+1	The queries and hash keys are different.
+2	The queries are different, but the hash keys are the same.
+10	The abstract plans are different.
100	One or both of the plan IDs does not exist.

- To find the ID of a plan, use sp_help_qpgroup or sp_find_qplan. Plan IDs are also returned by create plan and are included in showplan output.

Permissions

Any user can execute sp_cmp_qplans to compare plans that he or she owns. Only a System Administrator or the Database Owner can compare plans owned by another user.

See also

System procedures sp_cmp_all_qplans, sp_help_qpgroup

sp_commonkey

Description

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 the 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).

col1b
is the name of the partner column in the table or view *tabbname* that is joined with *col1a* in the table or view *tabaname*.

Examples

Example 1 Defines a common key on titles.titleid and titleauthor.titleid:

```
sp_commonkey titles, titleauthor, title_id, title_id
```

Example 2 Assumes two tables, projects and departments, each with a column named empid. This statement defines a frequently used join on the two columns:

```
sp_commonkey projects, departments, empid, empid
```

Usage

- 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. sp_commonkey does not enforce referential integrity constraints; use the primary key and foreign key clauses of the create table or alter table command to enforce key relationships.
- Executing sp_commonkey adds the key to the syskeys system table. To display a report on the common keys that have been defined, use 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 null types of the common columns need not agree.
- The installation process runs sp_commonkey on appropriate columns of the system tables.
- You cannot use a Java datatype with sp_commonnkey.

Permissions

Only the owner of *tabaname* or *tabbname* can execute sp_commonkey.

See also

Commands alter table, create table, create trigger

System procedures sp_dropkey, sp_foreignkey, sp_helpjoins, sp_helpkey, sp_primarykey

sp_companion

Description

Performs cluster operations such as configuring Adaptive Server as a secondary companion in a high availability system and moving a companion server from one failover mode to another. sp_companion is run from the secondary companion.

Syntax

```
sp_companion
        [server_name
        {, configure
            [, {with_proxydb | NULL}]
            [, srvlogin]
            [, server_password]
            [, cluster_login]
            [, cluspassword]]
        | drop
        | suspend
        | resume
        | prepare_failback
        | do_advisory}
            {, all
            | help
            | group attribute_name
            | base attribute_name}
```

Parameters

server_name
: is the name of the Adaptive Server on which you are performing a cluster operation.

configure
: configures the server specified by *server_name* as the primary companion in a failover configuration.

drop
: permanently drops a companion from failover configuration. After the command has completed, the servers are in single-server mode.

suspend
: temporarily removes the companions from a failover configuration. After the command is completed, the companions are in suspended mode.

resume
: reverses the suspend command and resumes normal companion mode between the companions.

prepare_failback
: prepare the secondary companion to relinquish the primary companion’s resources so it can failback.

do_advisory
verifies that the secondary companion is compatible for successfully performing the primary companion's functions during failover mode.

- all – causes do_advisory the investigate all the parameters.
- help – displays information and syntax about the do_advisory parameter.
- group attribute – limits do_advisory to investigate only the group attributes.
- base attribute – limits do_advisory to investigate only the base attributes.

with_proxydb
vreates proxy databases on the secondary companion for all database other than the system databases – and all subsequent databases that are added – when this parameter is included in the initial configuration of the companion servers. By default, with_proxydb is disabled.

srvlogin
is a user's login to access the companion server. By default, the value of srvlogin is "sa".

srvpassword
is the user's password to access the companion server. By default, the value of srvpassword is null.

cluster_login
i the user's login to log into the cluster. By default, the value of cluster_login is "sa".

cluspassword
is the users password you must provide to log into the cluster. By default, the value of cluspassword is null.

Examples

Example 1 Configures the Adaptive Server MONEY1 as the primary companion:

```
sp_companion "MONEY1", configure
```

Example 2 Configures the Adaptive Server MONEY1 as the primary companion and creates proxy databases on the secondary companion:

```
sp_companion "MONEY1", configure, with_proxydb, "sa", "sapsswd"
```

Example 3 Drops the Adaptive Server PERSONEL1 from the failover configuration. After the command has completed, both the primary companion and the secondary companion will be in single-server mode:

```
sp_companion "PERSONEL1", "drop"
```

Example 4 Resumes normal companion mode for the companion server (in this example, MONEY1):

```
sp_companion "MONEY1", "resume"
```

Example 5 Prepares the primary companion (in this example, PERSONEL1) to change to normal companion mode and resume control of the Adaptive Server that failed over:

```
sp_companion "PERSONEL1", "prepare_failback"
```

Example 6 Checks to make sure a cluster operation with the PERSONEL1 companion will be successful. Because do_advisory in this example uses the all parameter, it checks all the do_advisory attributes of PERSONEL1 to make sure that none of them will prevent a successful cluster operation, and that the secondary companion can successfully perform the primary companion's operations after failover is complete:

```
sp_companion "PERSONEL1", do_advisory, "all"
```

Example 7 Checks to make sure that none of the attributes for the Component Integration Services (CIS) on the companion server is compatible with the local server:

```
sp_companion "PERSONEL1", do_advisory, "CIS"
```

Usage

- sp_companion performs cluster operations such as configuring Adaptive Server as a secondary companion in a high availability system. sp_companion also moves companion servers from one failover mode to another (for example, from failover mode back to normal companion mode). sp_companion is run from the secondary companion.
- sp_companion is installed with the *installhasvss* (*insthasv* on Windows NT), not the *installmaster* script. *installhasvss* is located in *$SYBASE/ASE-12_0/scripts*.
- sp_companion automatically disables Sybase's mirroring. Sybase recommends that you use a third-party mirroring software to protect your data from disk failures.

For complete information, see *Using Sybase Failover in A High Availability System*. Before running the do_advisory command, make sure to read the configuration chapter of this book as well as the do_advisory chapter.

Permissions

Only users with the ha_role can issue sp_companion.

sp_configure

Description

Displays configuration parameters by group, their current values, their default values, the value to which they have most recently been set, and the amount of memory used by this setting. Displays only the parameters whose display level is the same as or below that of the user.

Syntax

sp_configure [*configname* [, *configvalue*] | *group_name* |
    non_unique_parameter_fragment][number of histogram steps, *n*]

sp_configure "configuration file", 0, {"write" | "read" | "verify" | "restore"}
    "*file_name*"

sp_configure “max concurrently recovered db”, *config_value*

sp_configure “number of checkpoint tasks”, *config_value*

Parameters

configname
displays the current value, default value, most recently changed value, and amount of memory used by the setting for all parameters matching *parameter*.

configvalue
resets *configname* to *configvalue* and displays the current value, default value, configured value, and amount of memory used by *configname*.

sp_configure *configname,* 0, “default” resets *configname* to its default value and displays current value, default value, configured value, and amount of memory used by *configname*.

group_name
displays all configuration parameters in *group_name*, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting.

non_unique_parameter_fragment
displays all parameter names that match *non_unique_parameter_fragment*, their current values, default values, configured values, and the amount of memory used.

file_name
is the name of the file you want to us sp_configure on.

number of histogram steps, n
used when creating an index or running update statistics where you can specify the number of steps when using create index or update statistics. The default is 20.

write
: creates *file_name* from the current configuration. If *file_name* already exists, a message is written to the error log and the existing file is renamed using the convention *file_name.001*, *file_name.002*, and so on. If you have changed a static parameter but have not restarted your server, “write” gives you the currently running value for that parameter.

read
: performs validation checking on values contained in *file_name* and reads those values that pass validation into the server. If any parameters are missing from *file_name*, the current running values for those parameters are used.

verify
: performs validation checking on the values in *file_name*.

restore
: creates *file_name* with the values in sysconfigures. This is useful if all copies of the configuration file have been lost and you need to generate a new copy.

max concurrently recovered db *config_value*
: sets the limit to the number of databases recovered concurrently.

  The default value of *config_value* is 0, where Adaptive Server determines the number of databases recovered concurrently, based on a self-tuning approach. The number of engines and number of open databases limit the value of this configuration parameter. There is no absolute maximum.The value of 1 indicates serial recovery.

number of checkpoint tasks *config_value*
: sets the limit to the number of checkpoint tasks in Adaptive Server.

  The default value of *config_value* is 1, indicating serial checkpoints. The number of engines and number of open databases limit number of checkpoint tasks. It has an absolute ceiling of 8.

Examples

Example 1 Displays all configuration parameters by group, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting:

```
sp_configure
```

Example 2 Displays all configuration parameters that include the word “identity”:

```
    sp_configure "identity"
Configuration option is not unique.
```

```
Parameter Name     Default Memory Used Config Value Run Value Unit Type
--------------     ------- ----------- ------------ --------- ------ ----
identity burning set     1           0            1         1     id    static
identity grab size       0           0            0         0     id    dyna
size of auto identit    10           0           10        10     bytes dyna
. . .
```

Example 3 Sets the system recovery interval in minutes to 3 minutes:

```
                    sp_configure "recovery interval in minutes", 3
Parameter Name      Default Memory Used Config Value Run Value Unit Type
--------------       ------- ----------- ------------ --------- ------ ----
recovery interval   5        0          3             3         min   dyn

Configuration option changed. The SQL Server need not be rebooted since the
option is dynamic.
```

Example 4 Resets the value for number of devices to the Adaptive Server default:

```
sp_configure "number of device", 0, "default"
```

Example 5 To reset the default number of steps for create index and update statistics:

```
sp_configure 'number of histogram steps', 30
```

Example 6 To configure 4 databases to be recovered concurrently, enter:

```
sp_configure "max concurrently recovered db", 4
```

Example 7 To configure Adaptive Server for the self-tuning approach, enter:

```
sp_configure "max concurrently recovered db", 0
```

Example 8 To configure Adaptive Server to have serial recovery, enter:

```
sp_configure "max concurrently recovered db", 1
```

Example 9 To start four checkpoint tasks, enter:

```
sp_configure "number of checkpoint tasks", 4
```

Example 10 To revert to serial checkpoints, enter:

```
sp_configure "number of checkpoint tasks", 1
```

Usage

- Any user can execute sp_configure to display information about parameters and their current values, but not to modify parameters. System Administrators can execute sp_configure to change the values of most configuration parameters. Only System Security Officers can execute certain parameters. These are listed under “Permissions” in this section.

- sp_configure allows you to specify the value for configuration paramters in unit specifiers. The unit specifiers are p or P for pages, m or M for megabytes, and g or G for gigabytes. If you do not specify a unit, and you are configuring a parameter that controls memory, Adaptive Server uses the logical page size for the basic unit.
- When you execute sp_configure to modify a dynamic parameter:
  - The configuration and run values are updated.
  - The configuration file is updated.
  - The change takes effect immediately.
- When you execute sp_configure to modify a static parameter:
  - The configuration value is updated.
  - The configuration file is updated.
  - The change takes effect only when you restart Adaptive Server.
- When issued with no parameters, sp_configure displays a report of all configuration parameters by group, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting:
  - The default column in the report displays the value Adaptive Server is shipped with. If you do not explicitly reconfigure a parameter, it retains its default value.
  - The memory used column displays the amount of memory used by the parameter at its current value in kilobytes. Some related parameters draw from the same memory pool. For instance, the memory used for stack size and stack guard size is already accounted for in the memory used for number of user connections. If you added the memory used by each of these parameters separately, it would total more than the amount actually used. In the memory used column, parameters that "share" memory with other parameters are marked with a hash mark (#).
  - The config_value column displays the most recent value to which the configuration parameter has been set with sp_configure.

    - The run_value column displays the value being used by Adaptive Server. It changes after you modify a parameter's value with sp_configure and, for static parameters, after you restart Adaptive Server. This is the value stored in syscurconfigs.value.

    Note If the server uses a case-insensitive sort order, sp_configure with no parameters returns a list of all configuration parameters and groups in alphabetical order with no grouping displayed.

- Each configuration parameter has an associated display level. There are three display levels:
    - The "basic" level displays only the most basic parameters. It is appropriate for very general server tuning.
    - The "intermediate" level displays parameters that are somewhat more complex, as well as showing you all the "basic" parameters. This level is appropriate for a moderately complex level of server tuning.
    - The "comprehensive" level displays all parameters, including the most complex ones. This level is appropriate for users who do highly detailed server tuning.

  The default display level is "comprehensive". Setting one of the other display levels lets you work with a subset of the configuration parameter, shortening the amount of information displayed by sp_configure.

  The syntax for showing your current display level is:

  ```
  sp_displaylevel
  ```

- For information on the individual configuration parameters, see the *System Administration Guide*.

max concurrently recovered db

This parameter determines the degree of parallelism during database recovery:

- When Adaptive Server is not in recovery, this configuration parameter takes effect statically. However, when Adaptive Server is in recovery, a System Administrator can force serial recovery dynamically.
- The effectiveness of max concurrently recovered db is dependent on the database layout and the performance of underlying I/O subsystem.

number of checkpoint tasks

This parameter configures parallel checkpoints:

- Parallel checkpoints depend on the layout of the databases and performance of underlying I/O sybsystems. Tune this parameter depending on the number of active databases and the ability of the I/O subsystem to handle writes.
- This configuration parameter is dynamic. When the value for this parameter is reduced, checkpoint tasks drain out, and when the value is increased, additional tasks are created.

Permissions

Any user can execute sp_configure to display information about parameters and their current values.

Only System Administrators can execute sp_configure to modify values for:

- max concurrently recovered db
- number of checkpoint tasks

Only System Administrators and System Security Officers can execute sp_configure to modify configuration parameters.

Only System Security Officers can execute sp_configure to modify values for:

- allow procedure grouping
- allow select on syscomments.text
- allow updates
- audit queue size
- auditing
- current audit table
- remote access
- suspend auditing when full
- systemwide password expiration

System Administrators can modify all other parameters.

See also

For more information on max concurrently recovered db and number of checkpoint tasks, see Chapter 27, “Backing up and Restoring User Databases,” in the *System Administration Guide*.

Commands set

System procedures sp_dboption, sp_displaylevel, sp_helpconfig, sp_monitorconfig

sp_copy_all_qplans

Description

Copies all plans for one abstract plan group to another group.

Syntax

sp_copy_all_qplans *src_group*, *dest_group*

Parameters

src_group
is the name of the source abstract plan group.

dest_group
is the name of the abstract plan group to which the plans are to be copied.

Examples

Copies all of the abstract plans in the dev_plans group to the ap_stdin group:

```
sp_copy_all_qplans dev_plans, ap_stdin
```

Usage

- The destination group must exist before you can copy plans into it. It may contain plans.
- sp_copy_all_qplans calls sp_copy_qplan for each plan in the source group. Each plan is copied as a separate transaction, so any problem that keeps sp_copy_all_qplans from completing does not affect the plans that have already been copied.
- sp_copy_qplan prints messages when it cannot copy a particular abstract plan. You also see these messages when running sp_copy_all_qplans.
- If the query text for a plan in the destination group exactly matches the query text in the source group and the user ID is the same, the plan is not copied, and a message giving the plan ID is sent to the user, but the copying process continues with the next plan in the source group.
- Copying a very large number of abstract plans can take considerable time, and also requires space on the system segment in the database and space to log the changes to the database. Use sp_spaceused to check the size of sysqueryplans, and sp_helpsegment for the system and logsegment to check the space available.

Permissions

Any user can execute sp_copy_all_qplans to copy an abstract plan that he or she owns. Only the System Administrator or Database Owner can copy plans that are owned by other users.

See also

System procedures sp_copy_qplan, sp_help_qpgroup

sp_copy_qplan

Description	Copies one abstract plan to an abstract plan group.
Syntax	sp_copy_qplan *src_id*, *dest_group*

Parameters

src_id
is the ID of the abstract plan to copy.

dest_group
is the name of the destination abstract plan group.

Examples

```
sp_copy_qplan 2140534659, ap_stdin
```

Usage

- The destination group must exist before you can copy an abstract plan into it. You do not need to specify a source group, since plans are uniquely identified by the plan ID.
- A new plan ID is generated when the plan is copied. The plan retains the ID of the user who created it, even if the System Administrator or Database Owner copies the plan. To assign a different user ID, a System Administrator or Database Owner can use sp_export_qpgroup and sp_import_qpgroup.
- If the query text for a plan in the destination group exactly matches the query text in the source group and the user ID, the plan is not copied, and a message giving the plan IDs is sent to the user.
- To copy all of the plans in an abstract plan group, use sp_copy_all_qplans.

Permissions

Any user can execute sp_copy_qplan to copy a plan that he or she owns. Only the System Administrator or Database Owner can copy plans that are owned by other users.

See also

System procedures sp_copy_all_qplans, sp_help_qpgroup, sp_help_qplan, sp_import_qpgroup

sp_countmetadata

Description

Displays the number of indexes, objects, or databases in Adaptive Server.

Syntax

sp_countmetadata "*configname*" [, *dbname*]

Parameters

configname
is either "open indexes", "open objects", or "open databases".

dbname
is the name of the database on which to run sp_countmetadata. If no database name is given, sp_countmetadata provides a total count for all databases.

Examples

Example 1 Reports on the number of user objects in Adaptive Server. Use this value to set the number of objects allowed in the database, plus space for additional objects and temporary tables:

```
sp_configure "number of open objects", 310
```

```
sp_countmetadata "open objects"
```

```
There are 283 user objects in all database(s), requiring
117.180 Kbytes of memory. The 'open objects'
configuration parameter is currently set to a run value
of 500.
```

Example 2 Reports on the number of indexes in Adaptive Server:

```
sp_countmetadata "open indexes", pubs2
```

```
There are 21 user indexes in pubs2 database(s),
requiring 8.613 kbytes of memory. The 'open indexes'
configuration parameter is currently set to 600.
```

Usage

- sp_countmetadata displays the number of indexes, objects, or databases in Adaptive Server, including the number of system databases such as model and tempdb.
- Avoid running sp_countmetadata during Adaptive Server peak times. It can cause contention on the sysindexes, sysobjects, and sysdatabases system tables.
- You can run sp_countmetadata on a specified database if you want information on a particular database. However, when configuring caches for indexes, objects, or databases, run sp_countmetadata without the *database_name* option.

- The information on memory returned by sp_countmetadata can vary by platform. For example, a database on Adaptive Server for Windows NT could have a different sp_countmetadata result than the same database on Sun Solaris. Information on the number of user indexes, objects, or databases should be consistent, however.
- sp_countmetadata does not include temporary tables in its calculation. Add 5 percent to the open objects value and 10 percent to the open indexes value to accommodate temporary tables.
- If you specify a nonunique fragment of "open indexes", "open objects", or "open databases" for *configname*, sp_countmetadata returns a list of matching configuration parameter names with their configured values and current values. For example:

```
sp_countmetadata "open"

Configuration option is not unique.
 option_name                    config_value run_value
 ------------------------------ ------------ -----------
 curread change w/ open cursors            1           1
 number of open databases                 12          12
 number of open indexes                  500         500
 number of open objects                  500         500
 open index hash spinlock ratio          100         100
 open index spinlock ratio               100         100
 open object spinlock ratio              100         100
```

Permissions

Only a System Administrator or the Database Owner can execute sp_countmetadata.

See also

System procedures sp_configure, sp_helpconfig, sp_monitorconfig

sp_cursorinfo

Description

Reports information about a specific cursor or all cursors that are active for your session.

Syntax

sp_cursorinfo [{*cursor_level* | null}] [, *cursor_name*]

Parameters

cursor_level | null

is the level at which Adaptive 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.

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. Adaptive Server reports information about all active cursors that use this name at the *cursor_level* you specify. If you omit this parameter, Adaptive Server reports information about all the cursors at that level.

Examples

Example 1 Displays the information about the cursor named authors_crsr at level 0:

```
sp_cursorinfo 0, authors_crsr
```

```
Cursor name 'authors_crsr' is declared at nesting level '0'.
The cursor id is 327681
The cursor has been successfully opened 1 times.
The cursor was compiled at isolation level 0.
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 read only.
There are 3 columns returned by this cursor.
The result columns are:
Name = 'au_id', Table = 'authors', Type = ID,
     Length = 11 (read only)
Name = 'au_lname', Table = 'authors', Type = VARCHAR,
Length = 40 (read only)
Name = 'au_fname', Table = 'authors', Type = VARCHAR,
Length = 20 (read only)
```

Example 2 Displays the information about any cursors named author_sales declared by a user across all levels:

```
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 id is 327682
The cursor has been successfully opened 1 times.
The cursor was compiled at isolation level 1.
The cursor is currently scanning at a nonzero isolation level.
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)
```

Usage

- If you do not specify either *cursor_level* or *cursor_name*, Adaptive Server displays information about all active cursors. Active cursors are those declared by you and allocated by Adaptive Server.
- Adaptive Server reports the following information about each cursor:
  - The cursor name, its nesting level, its cursor ID, and the procedure name (if it is declared in a stored procedure).
  - The number of times the cursor has been opened.
  - The isolation level (0, 1, or 3) in which it was compiled and in which it is currently scanning (if open).
  - 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 whether it is updatable.

The output from sp_cursorinfo varies, depending on the status of the cursor. In addition to the information listed, sp_cursorinfo displays the showplan output for the cursor. For more information about showplan, see the *Performance and Tuning Guide*.

Permissions

Any user can execute sp_cursorinfo.

See also

Commands declare cursor, set

sp_dbextend

Description

Allows you to:

- Install automatic database expansion procedures on database/segment pairs and devices.
- Define site-specific policies for individual segments and devices.
- Simulate execution of the database expansion machinery, to study the operation before engaging large volume loads.

These policies are stored in the sysattributes table in master database.

All arguments are string arguments:

Syntax

sp_dbextend ’help’[, <*command* >]

sp_dbextend [['set', ['threshold', *dbname*, *segmentname*, *freespace* |
    'database', *dbname*, *segmentname* {[[, *growby*] [, *maxsize*]]} |
    'device', *devicename* { [[, *growby*] [, *maxsize*]] }] |
    'clear', 'threshold', *dbname*, *segmentname*

sp_dbextend 'clear', 'database' [, *dbname* [, *segmentname*]]

sp_dbextend 'clear', 'device' [, *devicename*]

sp_dbextend 'modify', 'database', *dbname*, *segmentname*,
    { 'growby' | 'maxsize' }, *newvalue*

sp_dbextend 'modify', 'device', *devicename*, { ’growby’ | ' maxsize ' },
    newvalue

sp_dbextend { 'list' | 'listfull' } [, 'database' [, *dbname* [, *segmentname*
    [, *order_by_clause*]]]]

sp_dbextend { 'list' | 'listfull' } [, 'device' [, *devicename* [, *order_by_clause*]]]

sp_dbextend 'check', 'database' [, *dbname* [, *segmentname*]]

sp_dbextend { 'simulate' | 'execute' }, *dbname*, *segmentname* [, *iterations*]

sp_dbextend 'trace', {'on' | 'off' }

sp_dbextend 'reload [defaults]'

sp_dbextend { 'enable' | 'disable' }, 'database' [, *dbname* [, *segmentname*]]

sp_dbextend 'who' [, '<spid>' | 'block' | 'all']

Parameters

set

sets the threshold at which a database, segment, or device should fire. The arguments are:

- threshold – specifies the free space level at which to install the threshold on a specified database and segment.

  You should always specify *freespace* in size unit specifiers, such as megabytes. If you specify no size units, the value of *freespace* is treated as the number of kilobytes in the segment.

- *database* – specifies the name of the database/segment pair, the size by which to alter the database, and the maximum size of the database, at which the expansion process stops.

  growby – is the rate, in unit specifiers or percentage values, at which the database grows at each expansion attempt. *maxsize* is the maximum size of the segment, after which no further expansion occurs. Both are optional parameters.

- *device* – defines the growth rate and maximum size of a device, in unit specifiers or percentage values, at which the device can grow. *maxsize* in devices is subject to OS disk limitations.

clear

clears any previously set rules of expansion for a specified database and segment or for a specified device.

modify

modifies previously set site-specific policies, such as *growby* and *maxsize*, for a database and segment.

Use *newvalue* to specify the new value you set for automatic expansion.

list

lists briefly existing rules for a specified database, segment, or device, and presents the data from master.db.sysattributes in a readable format. Allows you to view rules on a per-database or per-device basis.

Presents the current rules in effect.

Use *order_by_clause* to generate listings in a different order from the default ordering of name, type.

listfull

lists fully the site-specific policy rules, and includes a comment column in the sysattributes table that displays a datetime stamp for when the rule was set, and when it was last modified.

check

examines current policies and verifies that they are consistent with the current space layout in each segment. If any policy settings appear redundant, ineffective, or incorrect, a warning message appears.

simulate

simulates executing the database or device expansion schemes executed at runtime, according to the set of current policies implemented by the set command.

iterations specifies the number of times you simulate the expansion.

execute

performs the actual database/segment, or device, expansion, using the current set of policies.

reload defaults

reinitializes sysattributes with the system-supplied defaults for *growby* and *maxsize* in all databases, segments, and devices, and reverts the databases or devices to the original default behavior.

help

provides help information for all command parameters, such as *set* or *list*, or help information for any single command.

trace

traces the threshold procedure execution logic in all expansion processes.

enable, disable

enables or disables the automatic expansion procedures on a specified database segment or device.

who

shows any active expansion processes running currently. ‘<spid>’ restricts the output for a particular spid. Use:

- *block* to show tasks that currently cause blocking of the expansion process.
- *all* to show all currently active tasks.

freespace

specifies the free space value at which the threshold procedure is installed on the specified segment. Always use size unit specifiers, such as megabytes, to specify *freespace*.

dbname

is the name of the database in which the threshold is being installed.

segmentname
is the segment contained in database *dbname*.

devicename
is the logical name of the affected device.

newvalue
specifies the new value you set for automatic expansion when you modify a policy for a database/segment pair or device.

order_by_clause
generates listings in a different order from the default ordering in the *list* command. The default order is name, type.

iterations
specifies the number of times an expansion is simulated or executed.

growby
specifies the rate, in unit specifiers or percentage values, at which a specified database segment or device grows each time the threshold procedures are attempted.

maxsize
is the maximum size of a segment/database pair or device, the size at which automatic expansion must stop.

Examples

Example 1 *set thresholds* – installs the space expansion threshold on a log segment in the database pubs2 at 100MB:

```
sp_dbextend 'set', 'thresh’, pubs2, logsegment, '100m'
```

Example 2 *set database* – installs a policy for the logsegment segment, at a growth rate of 100MB per expansion attempt:

```
sp_dbextend 'set', 'database', pubs2, logsegment, '100m'
```

Example 3 *set device* – expands this device until either the OS disk space limitation or the device size of 32GB is reached:

```
sp_dbextend 'set', 'device', pubs2-datadev1, '100m'
```

Example 4 *clear* – shows how to clear all space-expansion thresholds previously installed in pubs2, logsegment:

```
sp_dbextend 'clear', 'thresh', pubs2, logsegment
```

You can also the space-expansion threshold for segment dataseg1 in pubs2, installed at a free space of 200MB:

```
sp_dbextend 'clear', 'thresh', pubs2, dataseg1, '200m'
```

Example 5 *modify* – defines the rate of growth as 5% of current value, in each expansion attempt:

```
sp_dbextend 'modify', 'da', pubs2, logsegment, 'growby', '5%'
```

A command can fail when *maxsize* is not previously defined:

```
sp_dbextend 'modify', 'device', pubs2_log_dev, 'maxsize', '2.3g'
```

Example 6 *list* – lists briefly the rules for all databases and devices:

```
sp_dbextend 'list'
```

This lists rules for all databases with names similar to 'pubs%':

```
sp_dbextend 'list', 'database', 'pubs%'
```

Example 7 *listfull* – lists the rules for all databases and devices, including a comment column showing a datetime stamp:

```
sp_dbextend 'listfull'
```

Example 8 *simulate* – simulates an expansion twice, without tripping the thresholds:

```
sp_dbextend 'simulate', pubs2, logsegment, '2'
```

Example 9 *execute* – executes an automatic expansion procedure:

```
sp_dbextend 'execute', pubs2, logsegment
```

Example 10 *help* – obtains help for a specific command:

```
sp_dbextend help, 'set'
```

Usage

- You can only set one automatic expansion threshold on any given database/segment pair. If you try to install another instance of the threshold procedure, even at a different free space value, an error is raised.
- You cannot set system-supplied defaults, only modify them. After you modify system defaults you can reset them by re-running the *installdbextend* script, or by using the reload defaults command.
- To disallow any automatic growth in a particular segment, either specify 0 for *growby* or *maxsize*, or do not install the threshold procedure at all. If you specify NULL for this parameter, defaults to the system-specified default *growby* rate is used.
- *maxsize* is the maximum size of the segment at which the automatic expansion process stops, not the maximum size of the database.

- There is no system-specified maximum size for the default database. If no *maxsize* value is specified, the size of the database is limited only by the physical limitations of the database device.
- To turn off the automatic growth feature on a particular device, specify 0 for *growby* or *maxsize*. If you do not specify a value for *growby*, the default expansion rate is used.
- You can set *maxsize* to a value larger than the total amount of disk space available on the device, but actual expansion is limited to the available disk space at the time expansion is attempted.
- When you use this stored procedure to clear a threshold, *dbname* and *segmentname* are required arguments.
- When you use this stored procedure to clear a database, and provide no *dbname* and *segmentname*, all policy rules—that is, all the relevant rows in master.dbo.sysattributes—for the current database and all segments in it are deleted. This is a good way to reverse all settings to default and restart.
- When you use this stored procedure to clear a device, if you do not provide a value for *devicename*, no policy rules are cleared. You can clear out the policy rules for a single device by providing *devicename* or using "%" to clear policies for all devices.
- You can specify *dbname*, *devicename*, and *segmentname* using patterns, so that names whose patterns match the specified pattern are considered for the clear, enable, disable, and list operations.
- You must have set a value or property before you can modify it. modify fails if no value was previously set. *growby* and *maxsize* are modified to the new value specified by *newvalue*
- The new value specified in *newvalue* remains in effect throughout subsequent attempts to expand either the database or device. Even if *newvalue* is less than the current size of the database, segment, or device, the object does not shrink. *newvalue* specifies only future expansion, and does not affect current sizes.
- When you use list for a database and provide no *dbname* or *segmentname*, all the policy rules (that is, rows in master.dbo.sysattributes) for all segments in the current database are listed.
- Provide *dbname* and *segmentname* to obtain policy rules for individual databases and for the segments inside them.

- When you use list for a device name and provide no *devicename*, default policy rules for all devices are listed. You can filter this to list the policy rules for a single device by providing *devicename* or use pattern specifiers for the *devicename*.
- You can simulate the expansion of only one database/segment pair at a time. Both *dbname* and *segmentname* are required arguments. You cannot use wildcard patterns in *dbname* or *segmentname* for execute or simulate commands.
- The maximum size of a device is 32Gb.
- Use *reload* to re-initialize your databases and devices after using modify and simulate. *reload* deletes any existing rows in master.dbo.sysattributes that describe system default behavior, and loads new rows.
- *reload* does not delete user-specified policies.
- trace turns the trace facility on or off throughout the server. If trace is on, messages appear in the server error log when a threshold fires. Use trace only for troubleshooting.

Permissions

sa_role permission is needed to run the *installdbextend* script, and execute permission is granted to public only on sp_dbextend.

Any user can execute the list parameter. All other commands must be granted Database Owner or sa_role permissions on the specified database.

Commands such as clear, that allow pattern specifiers for the *dbname* argument, require sa_role privilege.

The following command parameters require sa_role privilege: simulate, execute, check, reload defaults, trace.

If the automatic expansion procedures are installed on a segment by a Database Owner without sa_role privilege, the devices do not expand, because the user cannot run the disk resize command. Sybase recommends that a user with sa_role privilege run the set threshold command when installing the threshold procedure.

See also

Stored procedures sp_dropthreshold, sp_modifythreshold

Commands alter database, create database, disk init, disk resize

sp_dboption

Description

Displays or changes database options.Enables the feature Asynchronous Log Service.

Syntax

sp_dboption [*dbname*, *optname*, {true | false}]

Parameters

dbname
is the name of the database in which the option is to be set. You must be using master to execute sp_dboption with parameters (that is, to change a database option). You cannot, however, change option settings in the master database.

optname
is the name of the option to be set. Adaptive 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 turn the option on, false to turn it off.

Examples

Example 1 Displays a list of the database options:

```
sp_dboption

Settable database options
 database_options
 ------------------------
 abort tran on log full
 allow nulls by default
 auto identity
 dbo use only
 ddl in tran
 identity in nonunique index
 no chkpt on recovery
 no free space acctg
 read only
 select into/bulkcopy/pllsort
 single user
 trunc log on chkpt
 trunc. log on chkpt.
 unique auto_identity index
```

Example 2 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:

```
use pubs2
go
```

```
master..sp_dboption pubs2, "read", true
go
checkpoint
go
```

Example 3 Makes the database pubs2 writable again:

```
pubs2..sp_dboption pubs2, "read", false
go
checkpoint
go
```

Example 4 Allows select into, bcp and parallel sort operations on tables in the pubs2 database. The select into string uniquely identifies the select into/bulkcopy option from among all available database options:

```
use pubs2
go
master..sp_dboption pubs2, "select into", true
go
checkpoint
go
```

Note Quotes are required around the option because of the embedded space.

Example 5 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:

```
use mydb
go
master..sp_dboption mydb, "auto identity", true
go
checkpoint
go
```

Example 6 Automatically includes an IDENTITY column in the mydb tables' index keys, provided these tables already have an IDENTITY column. All indexes created on the tables will be internally unique:

```
use master
go
sp_dboption mydb, "identity in nonunique index", true
go
use mydb
go
```

```
checkpoint
go
```

Example 7 Automatically includes an IDENTITY column with a unique, nonclustered index for new tables in the pubs2 database:

```
use master
go
sp_dboption pubs2, "unique auto_identity index", true
go
use pubs2
go
checkpoint
go
```

Example 8 Sets Asynchronous Log Service (ALS) in a specified database, enabling the ULC (User Log Cache) and the log writer threads.

```
sp_dboption "mydb", "async log service", true
```

Example 9 Disables ALS in a specified database.

```
sp_dboption "mydb", "async log service", false
```

Usage

- The master database option settings cannot be changed.
- To display a list of 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 no chkpt on recovery option disables the trunc log on chkpt option when both are set with sp_dboption for the same database. This conflict is especially possible in the tempdb database which has trunc log on chkpt set to on as the default.
- 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.

Asynchronous Log Service (ALS) options

- The ALS option is disabled by default.
- The ALS option cannot be enabled in system databases, such as master or model.

- The ALS option is persistent; once you enable ALS on a specified database, you can dump and reload the database without disabling ALS. To disable this feature, you must use sp_dboption to set the parameter to false.

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 SQL standards. 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 create table or alter table column definition. allow nulls by default true reverses this.
- While the auto identity option is set to true (on), 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 columns with the select * statement. To retrieve it, you must explicitly mention the column name, SYB_IDENTITY_COL, in the select list.

  To set the precision of the automatic IDENTITY column, use the size of auto identity column configuration parameter.

  Though you can set auto identity to true in tempdb, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column.

  For a report on indexes in a particular table that includes the IDENTITY column, execute sp_helpindex.
- While the dbo use only option is set to true (on), only the database's owner can use the database.

- When the ddl in tran option is set to true (on), 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 (DDL) 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 DDL 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 when the ddl in tran option is set to true:

  alter table – clauses other than partition and unpartition are allowed
  create default
  create index
  create procedure
  create rule
  create schema
  create table
  create trigger
  create view
  drop default
  drop index
  drop procedure
  drop rule
  drop table
  drop trigger
  drop view
  grant
  revoke

- The following commands cannot be used inside a user-defined transaction under any circumstances:

  alter database
  alter table...lock
  alter table...partition
  alter table...unpartition
  create database
  disk init

dump database
dump transaction
drop database
load transaction
load database
select into
truncate table
update statistics

In addition, system procedures that create temporary tables or change the master database cannot be used inside user-defined transactions.

- The identity in nonunique index option automatically includes an IDENTITY column in a table's index keys, so that all indexes created on the table are unique. This database option makes logically nonunique indexes internally unique, and allows these indexes to be used to process updatable cursors and isolation level 0 reads.

  The table must already have an IDENTITY column for the identity in nonunique index option to work, either from a create table statement or by setting the auto identity database option to true before creating the table.

  Use identity in nonunique index if you plan to use cursors and isolation level 0 reads on tables with nonunique indexes. A unique index ensures that the cursor will be positioned at the correct row the next time a fetch is performed on that cursor. If you plan to use cursors on tables with unique indexes and any isolation level, you may want to use the unique auto_identity index option.

  For a report on indexes in a particular table that includes the IDENTITY column, execute sp_helpindex.

- 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 are not recomputed for those segments.

- The no chkpt on recovery option is set to true (on) 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 to false (off), the default condition, a checkpoint record is added to a database after it is recovered when you restart Adaptive Server. This checkpoint, which ensures that the recovery mechanism will not be rerun unnecessarily, changes the sequence number and causes a subsequent load of the transaction log from the primary database to fail.

Setting this option to true (on) 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 cannot modify any data.
- Setting the select into/bulkcopy/pllsort option to true (on) enables the use of writetext, select into a permanent table, "fast" bulk copy into a table that has no indexes or triggers, using bcp or the bulk copy library routines, and parallel sort. A transaction log dump cannot recover these minimally logged operations, so dump transaction to a dump device is prohibited. After non-logged operations are completed, set select into/bulk copy/pllsort to false (off) and issue dump database.

  Issuing the dump transaction statement after unlogged changes have been made to the database with select into, bulk copy, or parallel sort 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/pllsort option to true in order to select into a temporary table, since tempdb is never recovered. The option need not be set to true 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 (single-user mode).

  You cannot set single user to true in a user database from within a stored procedure or while users have the database open. You cannot set single user to true for tempdb.
- The trunc log on chkpt option means that if the transaction log has more than 50 rows of committed transactions, the transaction log is truncated (the 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.

- When the unique auto_identity index option is set to true, it adds an IDENTITY column with a unique, nonclustered index to new tables. By default, the IDENTITY column is a 10-digit numeric datatype, but you can change this default with the size of auto identity column configuration parameter. As with auto identity, the IDENTITY column is not visible when you select all columns with the select * statement. To retrieve it, you must explicitly mention the column name, SYB_IDENTITY_COL, in the select list.

  If you need to use cursors or isolation level 0 reads with nonunique indexes, use the identity in nonunique index option.

  Though you can set unique auto_identity index to true in tempdb, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column with a unique index.

Permissions

Only a System Administrator or the Database Owner can execute sp_dboption with parameters to change database options. A user aliased to the Database Owner cannot execute sp_dboption to change database options. Any user can execute sp_dboption with no parameters to view database options.

See also

Documents See the *System Administration Guide* for more information on database options.

Commands checkpoint, select

System procedures sp_configure, sp_helpdb, sp_helpindex, sp_helpjoins

Utilities bcp

sp_dbrecovery_order

Description

Specifies the order in which user databases are recovered and lists the user-defined recovery order of a database or all databases.

Syntax

sp_dbrecovery_order
    [*database_name* [, *rec_order* [, force]]]

Parameters

database_name
The name of the database being assigned a recovery order or the database whose user-defined recovery order is to be listed.

rec_order
The order in which the database is to be recovered. A *rec_order* of -1 deletes a specified database from the user-defined recovery sequence.

force
allows the user to insert a database into an existing recovery sequence without putting it at the end.

Examples

Example 1 Makes the pubs2 database the first user database to be recovered following a system failure:

```
sp_dbrecovery_order pubs2, 1
```

Example 2 Inserts the pubs3 database into third position in a user-defined recovery sequence. If another database was initially in third position, it is moved to fourth position, and all databases following it are moved accordingly:

```
sp_dbrecovery_order pubs3, 3, force
```

Example 3 Removes the pubs2 database from the user-defined recovery sequence. Subsequently, pubs2 will be recovered after all databases with a user-specified recovery order have recovered:

```
sp_dbrecovery_order pubs2, -1
```

Example 4 Lists the current recovery order of all databases with a recovery order assigned through sp_dbrecovery_order:

```
sp_dbrecovery_order
```

Usage

- You must be in the master database to use sp_dbrecovery_order to enter or modify a user-specified recovery order. You can list the user-defined recovery order of databases from any database.
- To change the user-defined recovery position of a database, use sp_dbrecovery_order to delete the database from the recovery sequence, then use sp_dbrecovery_order to insert it into a new position.

- System databases are always recovered before user databases. The system databases and their recovery order are:

  master
  model
  tempdb
  sybsystemdb
  sybsecurity
  sybsystemprocs

- If no database is assigned a recovery order through sp_dbrecovery_order, all user databases are recovered in order, by database ID, after system databases.
- If *database_name* is specified, but no *rec_order* is given, sp_dbrecovery_order shows the user-defined recovery position of the specified database.
- If *database_name* is not specified, sp_dbrecovery_order lists the recovery order of all databases with a user-assigned recovery order.
- The order of recovery assigned through sp_dbrecovery_order must be consecutive, starting with 1 and containing no gaps between values. The first database assigned a recovery order must be assigned a *rec_order* of 1. If three databases have been assigned a recovery order of 1, 2, and 3, you cannot assign the next database a recovery order of 5.

Permissions

Only a System Administrator can execute sp_dbrecovery_order.

sp_dbremap

Description	Forces Adaptive Server to recognize changes made by alter database. Run this procedure only when instructed to do so by an Adaptive Server message.
Syntax	sp_dbremap *dbname*
Parameters	*dbname* is the name of the database in which the alter database command was interrupted.

Examples

An alter database command changed the database sample_db. This command makes the changes visible to Adaptive Server:

```
sp_dbremap sample_db
```

Usage

- If an alter database statement issued on a database that is in the process of being dumped is interrupted, Adaptive 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 Adaptive Server until the next restart.

Permissions

Only a System Administrator or Database Owner can execute sp_dbremap.

See also

Commands alter database, dump database, dump transaction

sp_defaultloc

Description

Component Integration Services only Defines a default storage location for objects in a local database.

Syntax

sp_defaultloc *dbname*, {"*defaultloc*"| NULL}
[, "*defaulttype*"]

Parameters

dbname
is the name of a database being mapped to a remote storage location. The database must already have been defined by a create database statement. You cannot map system databases to a remote location.

defaultloc
is the remote storage location to which the database is being mapped. To direct the server to delete an existing default mapping for a database, supply NULL for this parameter. The value of *defaultloc* must end in a period (.), as follows:

```
server.dbname.owner.
```

defaulttype
is one of the values that specify the format of the object named by *object_loc*.The valid values are as follows. Enclose the *defaulttype* value in quotes:

- table – indicates that the object named by *object_loc* is a table accessible to a remote server. This value is the default for *defaulttype*.
- view – indicates that the object named by *object_loc* is a view managed by a remote server, processed as a table.
- rpc – indicates that the object named by *object_loc* is an RPC managed by a remote server; processes the result set from the RPC as a read-only table.

Examples

Example 1 sp_defaultloc defines the remote storage location pubs.dbo. in the remote server named SYBASE. It maps the database pubs to the remote location. A `create table book1` statement would create a table named book1 at the remote location. A create existing table statement for bookN would require that pubs.dbo.bookN already exist at the remote location, and information about table bookN would be stored in the local table bookN:

```
sp_defaultloc pubs, "SYBASE.pubs.dbo.", "table" create
table pubs.dbo.book1 (bridges char(15))
```

Example 2 Removes the mapping of the database pubs to a remote location:

```
sp_defaultloc pubs, NULL
```

Example 3 Identifies the remote storage location wallst.nasdaq.dbo where "wallst" is the value provided for *server_name*, "nasdaq" is provided for *database*, and "dbo" is provided for *owner*. The RPC sybase must already exist at the remote location. A `create existing table sybase` statement would store information about the result set from RPC sybase in local table ticktape. The result set from RPC sybase is regarded as a read-only table. Inserts, updates and deletes are not supported for RPCs:

```
sp_defaultloc ticktape,"wallst.nasdaq.dbo.", "rpc"
create existing table sybase (bestbuy integer)
```

Usage

- sp_defaultloc defines a default storage location for tables in a local database. It maps table names in a database to a remote location. It permits the user to establish a default for an entire database, rather than issue an sp_addobjectdef command before every create table and create existing table command.
- When *defaulttype* is table, view, or rpc, the *defaultloc* parameter takes the form:

  ```
  server_name.dbname.owner.
  ```

  - Note that the *defaultloc* specification ends in a period (.).
  - *server_name* represents a server already added to sysservers by sp_addserver. The *server_name* parameter is required.
  - *dbname* might not be required. Some server classes do not support it.
  - *owner* should always be provided to avoid ambiguity. If it is not provided, the remote object actually referenced could vary, depending on whether the external login corresponds to the remote object owner.
- Issue sp_defaultloc before any create table or create existing table statement. When either statement is used, the server uses the sysattributes table to determine whether any table mapping has been specified for the object about to be created or defined. If the mapping has been specified, a create table statement directs the table to be created at the location specified by *object_loc*. A create existing table statement stores information about the existing remote object in the local table.
- If you issue sp_defaultloc on defaulttype view and then issue create table, Component Integration Services creates a new table, not a view, on the remote server.
- Changing the default location for a database does not affect tables that have previously been mapped to a different default location.

- After tables in the database have been created, all future references to tables in *dbname* (by select, insert, delete and update) are mapped to the correct location.

Permissions

Any user can execute sp_defaultloc.

See also

Commands create existing table, create table

System procedures sp_addobjectdef, sp_addserver, sp_helpserver

sp_depends

Description

Displays information about database object dependencies—the view(s), trigger(s), and procedure(s)—in the database that depend on a specified table or view, and the table(s) and view(s) in the database on which the specified view, trigger, or procedure depends.

Also displays information about table column dependencies—the index(s), default(s), check constraint(s), rule(s), and referential integrity constraint(s)—defined in either the column specified, if *column_name* is provided, or on all the columns in the table, if *column_name* is not provided.

Syntax

sp_depends *objname*[, *column_name*]

Parameters

objname
is the name of the table, view, Transact-SQL stored procedure, SQLJ stored procedure, SQLJ function, or trigger to be examined for dependencies. You cannot specify a database name. Use owner names if the object owner is not the user running the command and is not the Database Owner.

column_name
is the name of the column to be examined for dependencies.

Examples

Example 1 Lists the database objects that depend on the table sysobjects:

```
sp_depends sysobjects
```

Example 2 Lists the database objects that depend on the titleview view, and the database objects on which the titleview view depends:

```
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
```

Example 3 Lists the database objects that depend on the titles table owned by the user “mary”. The quotes are needed, since the period is a special character:

```
sp_depends "mary.titles"
```

Example 4 Shows the column-level dependencies for all columns of the sysobjects table:

```
sp_depends sysobjects

Things inside the current database that reference the object.
object                                    type
----------------------------------------- ----------------
dbo.sp_dbupgrade                          stored procedure
dbo.sp_procxmode                          stored procedure

Dependent objects that reference all columns in the table. Use sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not included
in this report.

Column           Type         Object Names or Column Names
---------------------- ------------ ------------------------------
cache            permission   column permission
ckfirst          permission   column permission
crdate           permission   column permission
deltrig          permission   column permission
expdate          permission   column permission
id               index        sysobjects (id)
id               logical RI   From syscolumns (id) To sysobjects (id)
id               logical RI   From syscomments (id) To sysobjects (id)
id               logical RI   From sysdepends (id) To sysobjects (id)
id               logical RI   From sysindexes (id) To sysobjects (id)
id               logical RI   From syskeys (depid) To sysobjects (id)
id               logical RI   From syskeys (id) To sysobjects (id)
id               logical RI   From sysobjects (id) To sysprocedures (id)
id               logical RI   From sysobjects (id) To sysprotects (id)
id               logical RI   sysobjects (id)
id               permission   column permission
indexdel         permission   column permission
instrig          permission   column permission
loginame         permission   column permission
name             index        ncsysobjects (name, uid)
name             permission   column permission
objspare         permission   column permission
schemacnt        permission   column permission
seltrig          permission   column permission
sysstat          permission   column permission
sysstat2         permission   column permission
type             permission   column permission
uid              index        ncsysobjects (name, uid)
uid              logical RI   From sysobjects (uid) To sysusers (uid)
uid              permission   column permission
updtrig          permission   column permission
userstat         permission   column permission
versionts        permission   column permission
```

Example 5 Shows more details about the column-level dependencies for the id column of the sysobjects table:

```
sp_depends sysobjects, id

Things inside the current database that reference the object.
object                                             type
------------------------------------               -------------
dbo.sp_dbupgrade                                   stored procedure
dbo.sp_procxmode                                   stored procedure
Dependent objects that reference column id.
Columns referenced in stored procedures, views or triggers are not included
in this report.
Type          Property     Object Names or Column Names
                           Also see/Use command
----------   ---------     ----------------------------------
                           ----------------------------------
index         index       sysobjects (id)
                          sp_helpindex, drop index,
                          sp_helpconstraint, alter table drop constraint
logical RI    primary     sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    foreign     From syskeys (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syscolumns (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysdepends (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysindexes (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syskeys (depid) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syscomments (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysobjects (id) To sysprotects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysobjects (id) To sysprocedures (id)
                          sp_helpkey, sp_dropkey
permission    permission  column permission
                          sp_helprotect, grant/revoke
```

Example 6 Shows the column-level dependencies for all columns of the user-created table, titles:

```
1> sp_depends titles

Things inside the current database that reference the object.
object                               type
-----------------------------------  ---------------
dbo.deltitle                          trigger
dbo.history_proc                      stored procedure
dbo.title_proc                        stored procedure
dbo.titleid_proc                      stored procedure
dbo.titleview                         view
dbo.totalsales_trig                   trigger

Dependent objects that reference all columns in the table. Use sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not included
in this report.

Column     Type         Object Names or Column Names
------     -----        ----------------------------------------
pub_id     logical RI   From titles (pub_id) To publishers (pub_id)
pubdate    default      datedflt
title      index        titleind (title)
title      statistics   (title)
title_id   index        titleidind (title_id)
title_id   logical RI   From roysched (title_id) To titles (title_id)
title_id   logical RI   From salesdetail (title_id) To titles (title_id)
title_id   logical RI   From titleauthor (title_id) To titles (title_id)
title_id   logical RI   titles (title_id)
title_id   rule         title_idrule
title_id   statistics   (title_id)
type       default      typedflt
```

Example 7 Shows more details about the column-level dependencies for the pub_id column of the user-created titles table:

```
sp_depends titles, pub_id

Things inside the current database that reference the object.
object                                  type
------------------------------------    ----------------
dbo.deltitle                            trigger
dbo.history_proc                        stored procedure
dbo.title_proc                          stored procedure
dbo.titleid_proc                        stored procedure
dbo.titleview                          view
dbo.totalsales_trig                    trigger
Dependent objects that reference column pub_id.
Columns referenced in stored procedures, views or triggers are not
included in this report.
Type          Property    Object Names or Column Names
                          Also see/Use command
----------    ---------   ---------------------
                          ---------------------------
logical RI    foreign     From titles (pub_id) To publishers (pub_id)
                          sp_helpkey, sp_dropkey
```

Usage

- Executing sp_depends lists all objects in the current database that depend on *objname*, and on which *objname* depends. 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 dependent on that object. References to objects outside the current database are not reported.
- Before you modify or drop a column, use sp_depends to determine if the table contains any dependent objects that could be affected by the modification. For example, if you modify a column to use a new datatype, objects tied to the table may need to be redefined to be consistent with the column's new datatype.
- The sp_depends procedure determines the dependencies by looking at the sysdepends table.

  If the objects were created out of order (for example, if a procedure that uses a view was created before the view was created), no rows exist in sysdepends for the dependencies, and sp_depends does not report the dependencies.
- 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 for the updated column indicate whether the stored procedure updates the object. The selected column indicates whether the object is being used for a read cursor or a data modification statement.
- sp_depends follows these Adaptive Server rules for finding objects:
  - If the user does not specify an owner name, and the user executing the command owns an object with the specified name, that object is used.
  - If the user does not specify an owner name, and the user does not own an object of that name, but the Database Owner does, the Database Owner's object is used.

    - If neither the user nor the Database Owner 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 with a different owner.
    - If both the user and the Database Owner own objects with the specified name, and the user wants to access the Database Owner'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 Database Owner must always be qualified with the owner's name, as in Example 3.
- SQLJ functions and SQLJ stored procedures are Java methods wrapped in SQL wrappers. See *Java in Adaptive Server Enterprise* for more information.
    - SQLJ functions and SQLJ stored procedures are database objects for which you can list dependencies. The only dependencies of SQLJ stored procedures and SQLJ functions are Java classes.
    - If *objname* is a SQLJ stored procedure or SQLJ function, sp_depends lists the Java class in the routine's external name declared in the create statement, not classes specified as the return type or datatypes in the parameter list.
    - SQLJ stored procedures and SQLJ functions can be listed as dependencies of other database objects.

Permissions

Any user can execute sp_depends.

See also

Commands create procedure, create table, create view, execute

System procedures sp_help

sp_deviceattr

Description

UNIX platforms only Changes the dsync setting of an existing database device file.

Syntax

sp_deviceattr *logicalname*, *optname*, *optvalue*

Parameters

logicalname
is the name of an existing database device. The device can be stored on either an operating system file or a raw partition, but the dsync setting is ignored for raw partitions.

optname
is the name of the setting to change. Currently, the only acceptable value for *optname* is dsync.

optvalue
can be either "true" or "false."

Examples

Sets dsync on for the device named "file_device1":

```
sp_deviceattr file_device1, dsync, true
```

Usage

- For database devices stored on UNIX files, dsync determines whether updates to the device take place directly on the storage media, or are buffered by the UNIX file system.

  When dsync is on, writes to the database device occur directly to the physical storage media, and Adaptive Server can recover data on the device in the event of a system failure.

  When dsync is off, writes to the database device may be buffered by the UNIX file system. The UNIX file system may mark an update as being completed, even though the physical media has not yet been modified. In the event of a system failure, there is no guarantee that requests to update data have ever taken place on the physical media, and Adaptive Server may be unable to recover the database.

- After using sp_deviceattr to change the dsync setting, you must reboot Adaptive Server before the change takes affect.

- dsync is always on for the master device file. You cannot change the dsync setting for a master device file with sp_deviceattr.

- The dsync value should be turned off only when the databases on the device need not be recovered after a system failure. For example, you may consider turning dsync off for a device that stores only the tempdb database.

- Adaptive Server ignores the dsync setting for devices stored on raw partitions; updates to those devices are never buffered, regardless of the dsync setting.
- dsync is not used on the Windows NT platform.

Permissions

The user executing sp_deviceattr must have permission to update the sysdevices table.

See also

System procedures sp_helpdevice

sp_diskdefault

Description

Specifies 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 *logicalname,* {defaulton | defaultoff}

Parameters

logicalname

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 designates the database device as a default database device; defaultoff designates that the specified database device is not a default database device.

Use defaulton after adding a database device to the system with disk init. Use defaultoff to change the default status of the master device (which is designated as a default device when Adaptive Server is first installed).

Examples

The master device is no longer used by create database or alter database for default storage of a database:

```
sp_diskdefault master, defaultoff
```

Usage

- 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 Adaptive Server, the master device 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 simplifies recovery of the master database.

- To find out which database devices are default database devices, execute sp_helpdevice.

Permissions

Only a System Administrator can execute sp_diskdefault.

See also

Commands alter database, create database, disk init

System procedures sp_helpdevice

sp_displayaudit

Description

Displays the status of audit options.

Syntax

```
sp_displayaudit ["procedure" | "object" | "login" | "database" | "global" |
          "default_object" | "default_procedure" [, "name"]]
```

Parameters

procedure

displays the status of audit options for the specified stored procedure or trigger. If you do not specify a value for *name*, sp_displayaudit displays the active audit options for all procedures and triggers in the current database.

object

displays the status of audit options for the specified table or view. If you do not specify a value for *name*, sp_displayaudit displays the active audit options for all tables and views in the current database.

login

displays the status of audit options for the specified user login. If you do not specify a value for *name*, sp_displayaudit displays the active audit options for all logins in the master database.

database

displays the status of audit options for the specified database. If you do not specify a value for *name*, sp_displayaudit displays the active audit options for all databases on the server.

global

displays the status of the specified global audit option. If you do not specify a value for *name*, sp_displayaudit displays the active audit options for all procedures and triggers in the current database.

default_object

displays the default audit options that will be used for any new table or view created on the specified database. If you do not specify a value for *name*, sp_displayaudit displays the default audit options for all databases with active default audit settings.

default_procedure

displays the default audit options that will be used for any new procedure or trigger created on the specified database. If you do not specify a value for *name*, sp_displayaudit displays the default audit options for all databases with active default audit settings.

name

is the information for the specified parameter, as described in the following table:

Parameter	**Value for *name***
procedure	Procedure or trigger name
object	Table or view name
login	User login
database	Database name
global	Global audit option
default_object	Database name
default_procedure	Database name

Examples

Example 1 Displays the status of each category and all auditing options when you do not specify a parameter:

```
sp_displayaudit

Procedure/Trigger    Audit Option   Value Database
 -----------------   -------------- ----- ---------------------
 dbo.sp_altermessage exec_procedure on    sybsystemprocs
 dbo.sp_help         exec_procedure on    sybsystemprocs
 dbo.sp_who          exec_procedure on    sybsystemprocs
No databases currently have default sproc/trigger auditing enabled.
No objects currently have auditing enabled.
No databases currently have default table/view auditing enabled.
No logins currently have auditing enabled.
No databases currently have auditing enabled.

Option Name                   Value
------------------------------ ------------------------------
adhoc                         off
dbcc                          off
disk                          off
errors                        off
login                         off
logout                        off
navigator_role                off
oper_role                     off
replication_role              off
rpc                           off
sa_role                       off
security                      off
sso_role                      off
```

Example 2 Displays the status of all procedure audit options when you do not specify a procedure name:

```
sp_displayaudit "procedure"

Procedure/Trigger    Audit Option   Value Database
 -----------------   -------------- ----- ---------------------
 dbo.sp_altermessage exec_procedure on    sybsystemprocs
 dbo.sp_help         exec_procedure on    sybsystemprocs
 dbo.sp_who          exec_procedure on    sybsystemprocs
```

Example 3 Displays only the status of the procedure when you specify a name for a procedure:

```
sp_displayaudit "procedure", "sp_who"

Procedure/Trigger Audit Option    Value Database
----------------- --------------- ----- ---------------------
dbo.sp_who         exec_procedure  on    sybsystemprocs
```

Example 4 Displays the status of all global audit options when you do not specify a global audit option:

```
sp_displayaudit "global"

Option Name                     Value
------------------------------ ------------------------------
adhoc                           off
dbcc                            off
disk                            off
errors                          off
login                           off
logout                          off
navigator_role                  off
oper_role                       off
replication_role                off
rpc                             off
sa_role                         off
security                        off
sso_role                        off
```

Usage

- sp_displayaudit displays the status of audit options.
- The following table shows the valid auditing options for each parameter:

Object type parameter	Valid auditing options
procedure	exec_procedure, exec_trigger

Object type parameter	Valid auditing options
object	delete, func_obj_access, insert, reference, select, update
login	all, cmdtext, table_access, view_access
database	alter, bcp, bind, create, dbaccess, drop, dump, func_dbaccess, grant, load, revoke, setuser, truncate, unbind
global	adhoc, dbcc, disk, errors, login, logout, navigator_role, oper_role, replication_role, rpc, sa_role, security, sso_role
default_object	delete, func_obj_access, insert, reference, select, update
default_procedure	exec_procedure, exec_trigger

- You cannot specify a value for name unless you first specify an object type parameter.

Permissions

Only a System Security Officer can execute sp_displayaudit.

See also

Documents See the *System Administration Guide* for information on setting up auditing.

System procedures sp_audit

Utilities bcp

sp_displaylevel

Description

Sets or shows which Adaptive Server configuration parameters appear in sp_configure output.

Syntax

sp_displaylevel [*loginame* [, *level*]]

Parameters

loginame
is the Adaptive Server login of the user for whom you want to set or show the display level.

level
sets the display level to one of the following:

- “basic” display level shows just the most basic configuration parameters. This level is appropriate for very general server tuning.
- “ intermediate” display level shows configuration parameters that are somewhat more complex, as well as all the “basic” level parameters. This level is appropriate for moderately complex server tuning.
- “ comprehensive” display level shows all configuration parameters, including the most complex ones. This level is appropriate for highly detailed server tuning.

Examples

Example 1 Shows the current display level for the user who invoked sp_displaylevel:

```
sp_displaylevel

The current display level for login 'sa' is 'comprehensive'.
```

Example 2 Shows the current display level for the user “jerry”:

```
sp_displaylevel jerry

The current display level for login 'jerry' is 'intermediate'.
```

Example 3 Sets the display level to “comprehensive” for the user “jerry”:

```
sp_displaylevel jerry, comprehensive

The display level for login 'jerry' has been changed to 'comprehensive'.
```

Usage

See the *System Administration Guide* for details about display levels and configuration parameters.

Permissions

Only a System Administrator can execute sp_displaylevel to set the display level for another user. Any user can execute sp_displaylevel to set and show his or her own display level.

See also

System procedures sp_configure

sp_displaylogin

Description

Displays information about a login account. Also displays information about the hierarchy tree above or below the login account when you so specify.

Syntax

sp_displaylogin [*loginame* [, expand_up | expand_down]]

Parameters

loginame

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.

expand_up

specifies that Adaptive Server display all roles in the role hierarchy that contain the loginame role.

expand_down

specifies that Adaptive Server display all roles in the role hierarchy that are contained by the loginame role.

Examples

Example 1 Displays information about your server login account:

```
sp_displaylogin
```

```
Suid: 1
Loginame: sa
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
    sa_role (default ON)
    sso_role (default ON)
    oper_role (default ON)
Locked: NO
Date of Last Password Change: Nov 16 1994 10:08AM
```

Example 2 Displays information about the login account "susanne". The information displayed varies, depending on the role of the user executing sp_displaylogin:

```
sp_displaylogin susanne
```

```
Suid: 12
Loginame: susanne
Fullname:
Default Database: pubs2
Default Language:
Auto Login Script:
Configured Authorization:
    supervisor (default OFF)
Locked: NO
Date of Last Password Change: May 12 1997 11:09AM
```

Example 3 Displays information about all roles containing the role of the login account "pillai". The information displayed varies, depending on the role of the user executing sp_displaylogin:

```
sp_displaylogin pillai, expand_up
```

Example 4 Displays the login security-related parameters configured for a login:

```
sp_displaylogin joe
```

```
Suid: 294
Loginame: joe
Fullname: Joseph Resu
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
    intern_role (default OFF)
Locked: NO
Date of Last Password Change: Nov 24 1998 3:46PM
Password expiration interval : 5
Password expired : NO
Minimum password length:4
Maximum failed logins : 10
Current failed logins : 3
```

Usage

- sp_displaylogin displays configured roles, so even if you have made a role inactive with the set command, it is displayed.
- If there are any login triggers associated with the login in question, they are listed after the `Auto Login Script` line. For more information, see "Row-level access control" in Chapter 11, "Managing User Permissions," of the *System Administration Guide*.
- When you use sp_displaylogin to get information about your own account, you do not need to use the *loginame* 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, default database, default language, and whether your account is locked.
- If you are a System Security Officer or System Administrator, you can use the *loginame* parameter to access information about any account.

Permissions

Only a System Administrator or a System Security Officer can execute sp_displaylogin with the *loginame* and expand parameters to get information about other users' login accounts. Any user can execute sp_displaylogin to get information about his or her own login account.

See also

Stored procedures sp_activeroles, sp_displayroles, sp_helprotect, sp_modifylogin

sp_displayroles

Description

Displays all roles granted to another role, or displays the entire hierarchy tree of roles in table format.

Syntax

sp_displayroles [*grantee_name* [, *mode*]]

Parameters

grantee_name

is the login name of a user whose roles you want information about, or the name of a role you want information about.

mode

is one of the following:

- expand_up – shows the role hierarchy tree for the parent levels
- expand_down – shows the role hierarchy tree for the child levels
- display_info – shows the login security-related parameters configured for the specified role

Examples

Example 1 Displays all roles granted to the user issuing the command:

```
sp_displayroles

Role Name
------------------------------
supervisor_role
```

Example 2 Displays all roles granted to supervisor_role:

```
sp_displayroles "supervisor_role"

Role Name
------------------------------
clerk
```

Example 3 Displays the active roles granted to login “susanne” and the roles below it in the hierarchy:

```
sp_displayroles susanne, expand_down

Role Name         Parent Role Name       Level
----------------- ---------------------- ------
supervisor_role   NULL                      1
clerk_role       supervisor_role          2
```

Example 4 Displays the active roles granted to intern_role and the roles above it in the hierarchy:

```
sp_displayroles "intern_role", expand_up
```

Example 5 Shows the login security-related parameters configured for the specified role:

```
sp_displayroles physician_role, "display_info"
```

```
Role name = physician_role
Locked : NO
Date of Last Password Change : Oct 31 1999 3:33PM
Password expiration interval = 5
Password expired : NO
Minimum password length = 4
Maximum failed logins = 10
Current failed logins = 3
```

Usage

- When you specify the optional parameter expand_up or expand_down all directly granted roles contained by or containing the specified role name are displayed.

Permissions

Only a System Administrator or a System Security Officer can execute sp_displayroles to display information on roles activated by any other user. Any user can execute sp_displayroles to see his or her own active roles.

See also

Documents See “User-Defined Login Security” in the *System Administration Guide* for more information.

Commands alter role, create role, drop role, grant, revoke, set

System procedures sp_activeroles, sp_displaylogin, sp_helprotect, sp_modifylogin

sp_dropalias

Description	Removes the alias user name identity established with sp_addalias.
Syntax	sp_dropalias *loginame*
Parameters	*loginame* is the name (in master.dbo.syslogins) of the user who was aliased to another user.

Examples

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:

```
sp_dropalias victoria
```

Usage

- 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.
- You cannot drop the alias of a user who owns objects in the database that were created in version 12.0 or later. You must drop the objects before dropping the login.

Permissions

Only the Database Owner or a System Administrator can execute sp_dropalias.

See also

System procedures sp_addalias, sp_adduser, sp_droplogin, sp_dropuser, sp_helpuser

sp_drop_all_qplans

Description
Deletes all abstract plans in an abstract plan group.

Syntax
sp_drop_all_qplans *name*

Parameters
name
is the name of the abstract plan group from which to drop all plans.

Examples

```
sp_drop_all_qplans dev_test
```

Usage

- To drop individual plans, use sp_drop_qplan.
- To see the names of abstract plan groups in the current database, use sp_drop_qpgroup.
- sp_drop_all_qplans silently drops all plans in the group that belong to the specified user, or all plans in the group, if it is executed by a System Administrator or Database Owner.

Permissions
Any user can execute sp_drop_all_qplans to drop plans that he or she owns. Only a System Administrator or Database Owner can drop plans owned by other users.

See also
System procedures sp_drop_qplan, sp_drop_qpgroup

sp_dropdevice

Description

Drops an Adaptive Server database device or dump device.

Syntax

sp_dropdevice *logicalname*

Parameters

logicalname
is the name of the device as listed in master.dbo.sysdevices.name.

Examples

Example 1 Drops the device named tape5 from Adaptive Server:

```
sp_dropdevice tape5
```

Example 2 Drops the database device named fredsdata from Adaptive Server. The device must not be in use by any database:

```
sp_dropdevice fredsdata
```

Usage

- The sp_dropdevice procedure drops a device from Adaptive 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 Adaptive Server. Use operating system commands to delete a file after using sp_dropdevice.

Permissions

Only a System Administrator can execute sp_dropdevice.

See also

Commands drop database

System procedures sp_addumpdevice, sp_helpdb, sp_helpdevice

sp_dropengine

Description

Drops an engine from a specified engine group or, if the engine is the last one in the group, drops the engine group.

Syntax

sp_dropengine *engine_number*, *engine_group*

Parameters

engine_number
is the number of the engine you are dropping from the group. Values are between 0 and a maximum equal to the number of configured online engines, minus one.

engine_group
is the name of the engine group from which to drop the engine.

Examples

This statement drops engine number 2 from the group called DS_GROUP. If it is the last engine in the group, the group is also dropped:

```
sp_dropengine 2, DS_GROUP
```

Usage

- sp_dropengine can be invoked only from the master database.
- If *engine_number* is the last engine in *engine_group*, Adaptive Server also drops *engine_group*.
- The *engine_number* you specify must exist in *engine_group*.

Permissions

Only a System Administrator can execute sp_dropengine.

See also

System procedures sp_addengine

sp_dropexeclass

Description

Drops a user-defined execution class.

Syntax

sp_dropexeclass *classname*

Parameters

classname
is the name of the user-defined execution class to be dropped.

Examples

This statement drops the user-defined execution class DECISION:

```
sp_dropexeclass 'DECISION'
```

Usage

- An execution class helps define the execution precedence used by Adaptive Server to process tasks. See the *Performance and Tuning Guide* for more information on execution classes and execution attributes.
- *classname* must not be bound to any client application, login, or stored procedure. Unbind the execution class first, using sp_unbindexeclass, then drop the execution class, using sp_dropexeclass.
- You cannot drop system-defined execution classes.

Permissions

Only a System Administrator can execute sp_dropexeclass.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_showexeclass, sp_unbindexeclass

sp_dropextendedproc

Description	Removes an extended stored procedure (ESP).
Syntax	sp_dropextendedproc *esp_name*
Parameters	*esp_name* is the name of the extended stored procedure to be dropped.
Examples	Removes xp_echo:

```
sp_dropextendedproc xp_echo
```

Usage

- sp_dropextendedproc must be executed from the master database.
- The *esp_name* is case sensitive. It must precisely match the name with which the ESP was created.

Permissions: Only a System Administrator can execute sp_dropextendedproc.

See also

Commands drop procedure

System procedures sp_addextendedproc, sp_freedll, sp_helpextendedproc

sp_dropexternlogin

Description

Component Integration Services only Drops the definition of a remote login previously defined by sp_addexternlogin.

Syntax

sp_dropexternlogin *remote_server* [, *login_name*] [, *role_name*]

Parameters

remote_server
is the name of the remote server from which the local server is dropping account access. The *remote_server* is known to the local server by an entry in the master.dbo.sysservers table.

login_name
is a login account known to the local server. If *login_name* is not specified, the current account is used. *login_name* must exist in the master.dbo.syslogins table.

role_name
is the Adaptive Server user's assigned role.

Examples

Example 1 Drops the definition of an external login to the remote server CIS1012 from "bobj". Only the "bobj" account and the "sa" account can add or modify a remote login for "bobj":

```
sp_dropexternlogin CIS1012, bobj
```

Example 2 Drops the definition of an external login to the remote server SSB from users with the sa_role:

```
sp_dropexternlogin SSB, NULL, sa_role
```

Usage

- sp_dropexternlogin drops the definition of a remote login previously defined to the local server by sp_addexternlogin.
- You cannot execute sp_dropexternlogin from within a transaction.
- The *remote_server* must be defined to the local server by sp_addserver.
- To add and drop local server users, use the system procedures sp_addalias and sp_droplogin.

Permissions

Only *login_name* or a System Administrator can execute sp_dropexternlogin.

See also

System procedures sp_addexternlogin, sp_helpexternlogin, sp_addlogin, sp_droplogin

sp_dropglockpromote

Description

Removes lock promotion values from a table or database.

Syntax

sp_dropglockpromote {"database" | "table"}, *objname*

Parameters

database | table
specifies whether to remove the lock promotion thresholds from a database or table. The quotes are required because these are Transact-SQL keywords.

objname
is the name of the table or database from which to remove the lock promotion thresholds.

Examples

Removes the lock promotion values from titles. Lock promotion for titles now uses the database or server-wide values:

```
sp_dropglockpromote "table", titles
```

Usage

- Use sp_dropglockpromote to drop lock promotion values set with sp_setpglockpromote.
- When you drop a database's lock promotion thresholds, tables that do not have lock promotion thresholds configured will use the server-wide values.
- When a table's values are dropped, Adaptive Server uses the database's lock promotion thresholds if they are configured or the server-wide values if they are not.
- Server-wide values can be changed with sp_setpglockpromote, but cannot be dropped.

Permissions

Only a System Administrator can execute sp_dropglockpromote.

See also

System procedures sp_setpglockpromote

sp_dropgroup

Description	Drops a group from a database.
Syntax	sp_dropgroup *grpname*
Parameters	*grpname* is the name of a group in the current database.

Examples

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:

```
sp_changegroup accounting, martha
sp_changegroup "public", george
sp_dropgroup purchasing
```

Usage

- 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.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute sp_dropgroup.

See also

System procedures sp_addgroup, sp_changegroup, sp_helpgroup

sp_dropkey

Description

Removes from the syskeys table a key that had been defined using sp_primarykey, sp_foreignkey, or sp_commonkey.

Syntax

sp_dropkey *keytype*, *tabname* [, *deptabname*]

Parameters

keytype
is the type of key to be dropped. The *keytype* must be primary, foreign, or common.

tabname
is the name of the key table or view that contains the key to be dropped.

deptabname
specifies the name of the second table in the relationship, 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

Example 1 Drops the primary key for the employees table. Any foreign keys that were dependent on the primary key for employees are also dropped:

```
sp_dropkey primary, employees
```

Example 2 Drops the common keys between the employees and projects tables:

```
sp_dropkey common, employees, projects
```

Example 3 Drops the foreign key between the titleauthor and titles tables:

```
sp_dropkey foreign, titleauthor, titles
```

Usage

- Executing sp_dropkey deletes the specified key from syskeys. Only the owner of a table can drop a key from 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.
- 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.

Permissions

Only the owner of *tabname* can execute sp_dropkey.

See also

System procedures sp_commonkey, sp_foreignkey, sp_helpkey, sp_primarykey

sp_droplanguage

Description

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 be dropped.

dropmessages
drops all Adaptive Server system messages in *language*. You cannot drop a language with associated system messages without also dropping its messages.

Examples

Example 1 This example drops French from the available alternate languages, if there are no associated messages:

```
sp_droplanguage french
```

Example 2 This example drops French from the available alternate languages, if there are associated messages:

```
sp_droplanguage french, dropmessages
```

Usage

- 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.

Permissions

Only a System Administrator can execute sp_droplanguage.

See also

System procedures sp_addlanguage, sp_helplanguage

sp_droplogin

Description

Drops an Adaptive Server user login by deleting the user's entry from master.dbo.syslogins.

Syntax

sp_droplogin *loginame*

Parameters

loginame
is the name of the user, as listed in master.dbo.syslogins.

Examples

Drops the "victoria" login from Adaptive Server:

```
sp_droplogin victoria
```

Usage

- Executing sp_droplogin drops a user login from Adaptive Server, deleting the user's entry from master.dbo.syslogins.
- Adaptive Server reuses a dropped login's server user ID, which compromises accountability. You can avoid dropping accounts entirely and, instead, use sp_locklogin to lock any accounts that will no longer be used.

  If you need to drop logins, be sure to audit these events (using sp_audit) so that you have a record of them.
- sp_droplogin deletes all resource limits associated with the dropped login.
- 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 account.

Permissions

Only a System Security Officer can execute sp_droplogin.

See also

System procedures sp_addlogin, sp_audit, sp_dropuser, sp_locklogin

sp_dropmessage

Description	Drops user-defined messages from sysusermessages.
Syntax	sp_dropmessage *message_num* [, *language*]

Parameters

message_num
is the message number of the message to be dropped. Message numbers must have a value of 20000 or higher.

language
is the language of the message to be dropped.

Examples

Removes the French version of the message with the number 20002 from sysusermessages:

```
sp_dropmessage 20002, french
```

Usage

- The *language* parameter is optional. If included, only the message with the indicated *message_num* in the indicated language is dropped. If you do not specify a *language*, all messages with the indicated *message_num* are dropped.

Permissions

Only the Database Owner, a System Administrator, or the user who created the message being dropped can execute sp_dropmessage.

See also

System procedures sp_addmessage, sp_getmessage

sp_dropobjectdef

Description

Component Integration Services only Deletes the external storage mapping provided for a local object.

Syntax

sp_dropobjectdef "*object_name*"

Parameters

object_name
has the form dbname.owner.object, where:

- *dbname* is the name of the database containing the object whose storage location you are dropping. *dbname* is optional; if present, it must be the current database, and the *owner* or a placeholder is required.
- *owner* is the name of the owner of the object whose storage location you are dropping. *owner* is optional; it is required if *dbname* is specified.
- *object* is the name of the local table for which external storage mapping is to be dropped.

Examples

Example 1 Deletes the entry from sysattributes that provided the external storage mapping for a table known to the server as the colleges table in database personnel:

```
sp_dropobjectdef "personnel.dbo.colleges"
```

Example 2 Deletes the entry from sysattributes that provided the external storage mapping for the andrea.fishbone object, where andrea is the owner and the local table name is fishbone:

```
sp_dropobjectdef "andrea.fishbone"
```

Usage

- sp_dropobjectdef deletes the external storage mapping provided for a local object. It replaces sp_droptabledef.
- Use sp_dropobjectdef after dropping a remote table with drop table.
- Dropping a table does not remove the mapping information from the sysattributes table if it was added using sp_addobjectdef. It must be explicitly removed using sp_dropobjectdef.
- The *object_name* can be in any of these forms:
  - *object*
  - *owner.object*
  - *dbname..object*
  - *dbname.owner.object*

Permissions

Only the Database Owner or a System Administrator can execute sp_dropobjectdef. Only a System Administrator can execute sp_dropobjectdef to remove mapping information for another user’s object.

See also

Commands create existing table, create table, drop table

System procedures sp_addobjectdef

sp_drop_qpgroup

Description

Drops an abstract plan group.

Syntax

sp_drop_qpgroup *group*

Parameters

group
is the name of the abstract plan group to drop.

Examples

Drops the abstract plan group “dev_test”:

```
sp_drop_qpgroup dev_test
```

Usage

- You cannot drop the default groups, ap_stdin and ap_stdout.
- You cannot drop a group that contains plans. To drop all of the plans in a a group, use sp_drop_all_qplans. To see a list of groups and the number of plans they contain, use sp_help_qpgroup.
- sp_drop_qpgroup cannot be run in a transaction.

Permissions

Only a System Administrator or Database Owner can execute sp_drop_qpgroup.

See also

System procedures sp_drop_all_qplans, sp_help_qpgroup

sp_drop_qplan

Description	Drops an abstract plan.
Syntax	sp_drop_qplan *id*
Parameters	*id* is the ID of the abstract plan to drop.

Examples

The abstract plan with the specified ID is dropped:

```
sp_drop_qplan 1760009301
```

Usage

- To find the ID of a plan, use sp_help_qpgroup, sp_help_qplan, or sp_find_qplan. Plan IDs are also returned by create plan and are included in showplan output.
- To drop all abstract plans in a group, use sp_drop_all_qplans.

Permissions

Any user can execute sp_drop_qplan to drop a plan he or she owns. Only the System Administrator or the Database Owner can drop plans owned by other others.

See also

Commands create plan

System procedures sp_drop_all_qplans, sp_find_qplan, sp_help_qpgroup, sp_help_qplan

sp_dropremotelogin

Description

Drops a remote user login.

Syntax

sp_dropremotelogin *remoteserver* [, *loginame* [, *remotename*]]

Parameters

remoteserver
is the name of the server that has the remote login to be dropped.

loginame
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 *loginame* when logging in from the remote server.

Examples

Example 1 Drops the entry for the remote server named GATEWAY:

```
sp_dropremotelogin GATEWAY
```

Example 2 Drops the entry for mapping remote logins from the remote server GATEWAY to the local user named "churchy":

```
sp_dropremotelogin GATEWAY, churchy
```

Example 3 Drops the login for the remote user "pogo" on the remote server GATEWAY that was mapped to the local user named "churchy":

```
sp_dropremotelogin GATEWAY, churchy, pogo
```

Usage

- 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.

Permissions

Only a System Administrator can execute sp_dropremotelogin.

See also

System procedures sp_addlogin, sp_addremotelogin, sp_addserver, sp_droplogin, sp_helpremotelogin, sp_helpserver

sp_drop_resource_limit

Description

Removes one or more resource limits from Adaptive Server.

Syntax

```
sp_drop_resource_limit { name, appname }
          [, rangename, limittype, enforced, action, scope]
```

Parameters

name

is the Adaptive Server login to which the limit applies. To drop resource limits that apply to all users of a particular application, specify the *appname* and a *name* of NULL.

appname

is the application to which the limit applies. To drop resource limits that apply to all applications used by the specified login, specify the login name and an *appname* of NULL. To drop a limit that applies to a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename

is the time range during which the limit is enforced. This must be an existing time range stored in the systimeranges system table or NULL to delete all resource limits for the specified *name*, *appname*, *limittype*, *action*, and *scope*, without regard to *rangename*.

limittype

is the type of resource being limited. This must be one of the following:

- row_count – drops only limits that restrict the number of rows a query can return.
- elapsed_time – drops only limits that restrict the number of seconds that a query batch or transaction can run.
- io_cost – drops only limits that restrict actual or estimated query processing cost.
- tempdb_space – drops only limits that restrict the number of pages a tempdb database can have during a single session.
- NULL – drops all resource limits with the specified *name*, *appname*, *rangename*, enforcement time, *action*, and *scope*, without regard to *limittype*.

enforced

determines whether the limit is enforced prior to or during query execution. The following table lists the valid values for each limit type:

Enforced code	Description	Limit type
1	Drops only limits for which action is taken when the estimated cost of execution exceeds the specified limit.	io_cost
2	Drops only limits for which action is taken when the actual row count, elapsed time, or cost of execution exceeds the specified limit.	row_count elapsed_time io_cost
3	Drops only limits for which action is taken when either the estimated cost (1) or the actual cost (2) exceeds the specified limit.	io_cost
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, and *scope*, without regard to when the *action* is enforced.	

action

is the action taken when the limit is exceeded. This must be one of the following:

Action code	Description
1	Drops only limits that issue a warning.
2	Drops only limits that abort the query batch.
3	Drops only limits that abort the transaction.
4	Drops only limits that kill the session.
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, enforcement time, and *scope*, without regard to the *action* they take.

scope

is the scope of the limit. This must be one of the following:

Scope code	Description
1	Drops only limits that apply to queries.
2	Drops only limits that apply to query batches.
4	Drops only limits that apply to transactions.
6	Drops only limits that apply to both query batches and transactions.
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, enforcement time, and *action*, without regard to their *scope*.

Examples

Example 1 Drops the single resource limit that kills the session whenever joe's use of the *payroll* application runs a query during the *friday_afternoon* time range that results in excessive execution-time I/O cost:

```
sp_drop_resource_limit joe, payroll, friday_afternoon, io_cost, 2, 4, 1
```

Note If no resource limit matches these selection criteria, sp_drop_resource_limit returns without error.

Example 2 Drops all limits that apply to joe's use of the *payroll* application:

```
sp_drop_resource_limit joe, payroll
```

Example 3 Drops all limits that apply to the user "joe":

```
sp_drop_resource_limit joe
```

Example 4 Drops all resource limits that apply to the *payroll* application:

```
sp_drop_resource_limit NULL, payroll
```

Example 5 Drops all resource limits on the *payroll* application whose action is to kill the session:

```
sp_drop_resource_limit NULL, payroll, NULL, NULL, NULL, 4, NULL
```

Usage

- Use the sp_help_resource_limit system procedure to determine which resource limits apply to a given user, application, or time of day.
- When you use sp_droplogin to drop an Adaptive Server login, all resource limits associated with that login are also dropped.
- The deletion of a resource limit causes the limits for each session for that login and/or application to be rebound at the beginning of the next query batch for that session.

Permissions

Only a System Administrator can execute sp_drop_resource_limit.

See also

Documents See the *System Administration Guide* for more information on resource limits.

System procedures sp_add_resource_limit, sp_droplogin, sp_help_resource_limit, sp_modify_resource_limit

sp_droprowlockpromote

Description

Removes row lock promotion threshold values from a database or table.

Syntax

sp_droprowlockpromote {"database" | "table"}, *objname*

Parameters

database | table

specifies whether to remove the row lock promotion thresholds from a database or table.

objname

is the name of the database or table from which to remove the row lock promotion thresholds.

Examples

Removes the row lock promotion values from the sales table. Lock promotion for sales now uses the database or server-wide values:

```
sp_droprowlockpromote "table", "sales"
```

Usage

- Use sp_droprowlockpromote to drop row lock promotion values set with sp_setrowlockpromote.
- When you drop a database's row lock promotion thresholds, datarows-locked tables that do not have row lock promotion thresholds configured use the server-wide values. Use sp_configure to check the value of the row lock promotion configuration parameters.
- When a table's row lock promotion values are dropped, Adaptive Server uses the database's row lock promotion thresholds, if they are configured, or the server-wide values, if no thresholds are set for the database.
- To change the lock promotion thresholds for a database, you must be using the master database. To change the lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- Server-wide values can be changed with sp_setrowlockpromote. This changes the values in the row lock promotion configuration parameters, so there is no corresponding server option for sp_droprowlockpromote.

Permissions

Only a System Administrator can execute sp_droprowlockpromote.

See also

System procedures sp_setrowlockpromote

sp_dropsegment

Description

Drops a segment from a database or unmaps a segment from a particular database device.

Syntax

sp_dropsegment *segname*, *dbname* [, *device*]

Parameters

segname
is the name of the segment to be dropped.

dbname
is the name of the database from which the segment is to be dropped.

device
is the name of the database device from which the segment *segname* is to be dropped. This parameter is optional, except when the system segment system, default, or logsegment is being dropped from a database device.

Examples

Example 1 This command drops the segment indexes from the pubs2 database.

```
sp_dropsegment indexes, pubs2
```

Example 2 This command unmaps the segment indexes from the database device dev1:

```
sp_dropsegment indexes, pubs2, dev1
```

Usage

- 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 *device*, the segment is dropped from the specified database. If you do supply a *device* 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, Adaptive Server drops any thresholds that exceed the total space on the segment. When you unmap the logsegment from one or more devices, Adaptive Server recalculates the last-chance threshold.
- sp_placeobject changes future space allocations for a table or index from one segment to another, and removes the references from the original segment. After using sp_placeobject, 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.

Permissions	Only the Database Owner or a System Administrator can execute sp_dropsegment.
See also	**System procedures** sp_addsegment, sp_addthreshold, sp_helpsegment, sp_helpthreshold, sp_placeobject

sp_dropserver

Description

Drops a server from the list of known servers or drops remote logins and external logins in the same operation.

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

Example 1 This command drops the remote server GATEWAY:

```
sp_dropserver GATEWAY
```

Example 2 Drops the entry for the remote server RDBAM_ALPHA and drops all remote logins and external logins for that server:

```
sp_dropserver RDBAM_ALPHA, droplogins
```

Usage

- Executing sp_dropserver drops a server from the list of known servers by 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 remote logins for a server when dropping the server, use droplogins.
- Running sp_dropserver without droplogins against a server that has associated entries in the sysattributes table results in an error. You must drop the remote logins and external logins before you can drop the server.
- The checks against sysattributes for external logins and for default mapping to a server apply when Component Integration Services is configured.

Permissions

Only a System Security Officer can execute sp_dropserver.

See also

System procedures sp_addserver, sp_dropremotelogin, sp_helpremotelogin, sp_helpserver

sp_dropthreshold

Description

Removes a free-space threshold from a segment.

Syntax

sp_dropthreshold *dbname*, *segname*, *free_space*

Parameters

dbname
is the database from which you are dropping the threshold. This must be the name of the current database.

segname
is the segment whose free space is monitored by the threshold. Use quotes when specifying the “default” segment.

free_space
is the number of free pages at which the threshold is crossed.

Examples

Removes a threshold from segment1 of mydb. You must specify the database, segment, and amount of free space to identify the threshold:

```
sp_dropthreshold mydb, segment1, 200
```

Usage

- 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.

Permissions

Only the Database Owner or a System Administrator can execute sp_dropthreshold.

See also

System procedures sp_addthreshold, sp_dboption, sp_helpthreshold, sp_thresholdaction

sp_drop_time_range

Description — Removes a user-defined time range from Adaptive Server.

Syntax — sp_drop_time_range *name*

Parameters

name
is the name of the time range to be dropped.

Examples — Removes the "evenings" time range:

```
sp_drop_time_range evenings
```

Usage

- You cannot remove the "at all times" time range.
- You cannot drop a time range if a resource limit exists for that time range.
- Dropping a time range does not affect the active time ranges for sessions currently in progress.

Permissions — Only a System Administrator can execute sp_drop_time_range.

See also

Documents For more information on time ranges, see the *System Administration Guide*.

System procedures sp_add_resource_limit, sp_add_time_range, sp_modify_time_range

sp_droptype

Description	Drops a user-defined datatype.
Syntax	sp_droptype *typename*
Parameters	*typename* is the name of a user-defined datatype that you own.

Examples

Drops the user-defined datatype named birthday:

```
sp_droptype birthday
```

Usage

- Executing sp_droptype deletes a user-defined datatype from systypes.
- A user-defined datatype cannot be dropped if it is referenced by tables or another database object.

Permissions

Only the Database Owner or datatype owner can execute sp_droptype.

See also

Datatypes User-defined datatypes

System procedures sp_addtype, sp_rename

sp_dropuser

Description	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

Drops the user "albert" from the current database. The user "albert" can no longer use the database:

```
sp_dropuser albert
```

Usage

- sp_dropuser drops a user from the current database by deleting the user's row from sysusers.
- You cannot drop a user who owns objects in the database.
- You cannot drop a user who has 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.
- You cannot drop a user from a database if the user owns a stored procedure that is bound to an execution class in that database. See sp_bindexeclass.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute sp_dropuser.

See also

Commands grant, revoke, use

System procedures sp_addalias, sp_adduser, sp_bindexeclass, sp_droplogin

sp_dumpoptimize

Description

Specifies the amount of data dumped by Backup Server during the dump database operation.

Syntax

```
sp_dumpoptimize [ 'archive_space =
          {maximum | minimum | default }' ]
```

```
sp_dumpoptimize [ 'reserved_threshold =
          {nnn | default }' ]
```

```
sp_dumpoptimize [ 'allocation_threshold =
          {nnn | default }' ]
```

Parameters

archive_space
: specifies the amount of the database you want dumped.

maximum
: dumps the whole database without determining which pages are allocated or not. The total space used by the archive image or images is equal to the size of the database. Using this option has the same effect as using the options reserved_threshold=0 and allocation_threshold=0.

minimum
: dumps only the allocated pages, which results in the smallest possible archive image. This option is useful when dumping to archive devices for which the throughput is much smaller than that of the database devices such as QIC tape drives. Using this option has the same effect as using the options reserved_threshold=100 and allocation_threshold=100.

default
: specifies that default values should be used.

When used with archive_space, this option dumps the database with the reserved_threshold and allocation_threshold options set to their default values. Use this to reset Backup Server to the default configuration.

When used with reserved_threshold, default specifies 85 percent.

When used with allocation_threshold, default specifies 40 percent.

reserved_threshold
: dumps all the pages belonging to the database in a database disk if the percentage of reserved pages in the disk is equal to or greater than *nnn*. For example, if you specify *nnn* as 60 and if a database disk has a percentage of reserved pages equal to or greater than 60 percent, then the entire disk is dumped without determining which pages within that disk are allocated. The default for this option is 85 percent.

nnn

an integer value between 0 and 100 that represents the value of the threshold. It is used to determine how much data to dump.

When used with reserved_threshold, if the percentage of reserved pages in the disk is greater than the value specified, all the pages of the database in a database disk are dumped.

When used with allocation_threshold, if the percentage of allocated pages in an allocation unit is greater than the percentage specified for allocation_threshold, all the pages within an allocation unit are dumped.

allocation_threshold

dumps all the pages in the allocation unit if the percentage of allocated pages in the unit is equal to or greater than *nnn*. For example, if *nnn* is specified as 70 and if the percentage of allocated pages in an allocation unit is equal to or greater than 70 percent, then the entire allocation unit is dumped without determining whether pages within that allocation unit are allocated or not. If the reserved_threshold setting causes the whole disk to be dumped, the allocation_threshold setting is ignored for the disk. The default for this option is 40 percent.

Examples

Example 1 This causes the whole database to be dumped:

```
sp_dumpoptimize 'archive_space=maximum'
```

```
Backup Server: 4.172.1.1: The value of 'reserved pages threshold' has been
set to 0%.
Backup Server: 4.172.1.2: The value of 'allocated pages threshold' has been
set to 0%.
```

Example 2 This causes only the allocated pages to be dumped, thereby resulting in the smallest archive image:

```
sp_dumpoptimize 'archive_space=minimum'
```

```
Backup Server: 4.172.1.1: The value of 'reserved pages threshold' has been
set to 100%.
Backup Server: 4.172.1.2: The value of 'allocated pages threshold' has been
set to 100%.
```

Example 3 This causes the reserved threshold to be set to 85 percent and the allocation threshold to be set to 40 percent:

```
sp_dumpoptimize 'archive_space=default'
```

```
Backup Server: 4.172.1.1: The value of 'reserved pages threshold' has been
set to 85%.
Backup Server: 4.172.1.2: The value of 'allocated pages threshold' has been
set to 40%.
```

Example 4 Those disks in the database whose percentage of reserved pages is greater than or equal to 60 percent are dumped without reading allocation pages on this disk. For the remaining disks, the allocation pages are read, and the last set value for the allocation_threshold is used. If the allocation_threshold was not set after Backup Server was started, default allocation_threshold of 40 percent is used:

```
sp_dumpoptimize 'reserved_threshold=60'
```

```
Backup Server: 4.172.1.3: The value of 'reserved pages threshold' has been
set to 60%.
```

Example 5 This causes the reserved threshold to be set to 85 percent. It does not affect the allocation page threshold:

```
sp_dumpoptimize 'reserved_threshold=default'
```

```
Backup Server: 4.172.1.3: The value of 'reserved pages threshold' has been
set to 85%.
```

Example 6 Allocation pages are read for those disks whose reserved page percentage is less than the last set value for the reserved_threshold and if an allocation unit has 80 percent or more pages allocated, then the whole allocation unit is dumped:

```
sp_dumpoptimize 'allocation_threshold=80'
```

```
Backup Server: 4.172.1.4: The value of 'allocated pages threshold' has been
set to 80%.
```

Example 7 This causes the allocation page threshold to be set to the default of 40 percent. It does not affect the reserved pages threshold:

```
sp_dumpoptimize 'allocation_threshold=default'
```

```
Backup Server: 4.172.1.4: The value of 'allocated pages threshold' has been
set to 40%.
```

Example 8 Those disks in the database whose percentage of reserved pages is greater than or equal to 60 percent are dumped without reading allocation pages on this disk. For the remaining disks, the allocation pages are read and if an allocation unit has 30 percent or more pages allocated, then the whole allocation unit is dumped:

```
sp_dumpoptimize 'reserved_threshold=60', 'allocation_threshold=30'
```

```
Backup Server: 4.172.1.3: The value of 'reserved pages threshold' has been
set to 60%.
Backup Server: 4.172.1.4: The value of 'allocated pages threshold' has been
set to 30%.
```

Example 9 This displays the current value of the thresholds:

```
sp_dumpoptimize
```

```
Backup Server: 4.171.1.1: The current value of 'reserved pages threshold'
is 60%
Backup Server: 4.171.1.2: The current value of 'allocated pages threshold'
is 30%.
```

Usage

- When you set values with sp_dumpoptimize, those values are immediately in affect without the need to restart Backup Server. However, the changes are effective only until the Backup Server is restarted. When Backup Server is restarted, the default values are used.
- If you issue sp_dumpoptimize multiple times, the thresholds specified by the last instance are used by later dumps. For example, if you first set the reserved_threshold value, and later issue archive_space=maximum, then that value overwrites the previous value you set for reserved_threshold.
- Dumps of different databases can use different thresholds by changing the sp_dumpoptimize values before each database dump.
- The optimal threshold values can vary from one database to another. Therefore, the performance of a dump depends on both the I/O configuration and the amount of used space in the database. The DBA can determine the appropriate configuration for a database by experimenting with dumps using different values and choosing the one that results in the shortest dump time.
- You can use sp_dumpoptimize for both local and remote dumps.
- sp_dumpoptimize has no effect on the performance of a transaction log dump or a load. Therefore, it need not be issued before dump transaction, load database or load transaction operations.
- If sp_dumpoptimize is issued without any parameters, the current value of the thresholds is displayed on the client.
- On configurations in which the archive device throughput is equal to or higher than the cumulative throughput of all the database disks, using archive_space=maximum may result in a faster dump. However, on configurations in which the archive device throughput is less than the cumulative throughput of all the database disks, using this option may result in a slower dump.
- The option names and the values for this procedure can be abbreviated to the unique substring that identifies them. For example, ar = ma is sufficient to uniquely identify the option archive_space=maximum.

- There can be zero or more blank space characters around the equal sign (=) in the option string.
- The option names and their values are case insensitive.

Permissions

Only the System Administrator, the Database Owner, or users with the Operator role can execute sp_dumpoptimize.

See also

Documents See the *System Administration Guide* for information on allocation pages.

Commands dump database, dump transaction, load database, load transaction

sp_engine

Description

Enables you to bring an engine online or offline.

Syntax

sp_engine {"online" | [offline | can_offline] [, *engine_id*] |
["shutdown", *engine_id*]}

Parameters

"online"
bring an engine online. The value of sp_configure "max online engines" must be greater than the current number of engines online. , Becuase "online" is a reserved keyword, you must use quotes.

offline
bring an engine offline. You can also use the *engine_id* parameter to specify a specific engine to bring offline.

can_offline
returns information on whether an engine can be brought offline. can_offline returns the Adaptive Server tasks with an affinity to this engine (for example, during Omni or java.net tasks) if its state is online . If you do not specify an *engine_id*, the command describes the status of the engine in sysengines with the highest *engine_id*.

engine_id
the ID of the engine. The *engine_id* parameter is optional. If you do not specify an *engine_id*, sp_engine uses the incremented or decremented value for *engine_id* for the value of engine found within sysengines. That is, if your system uses engines 0, 1, 2, and 3, and you do not specify an engine id, sp_engine takes engine ID 3 offline, then engine ID 2, and so on.

"shutdown"
Forces an engine offline. If there are any tasks with an affinity to this engine, they are killed after a five-minute wait. You must use quotes, as shutdown is a reserved keyword.

Examples

Example 1 Brings engine 1 online. Messages are platform specific (in this example, Sun Solaris was used):

```
sp_engine "online", 1
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device connection
limit is 3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security modules
loaded successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 2, os pid 8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel
asynchronous disk I/O strategy
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck: Network fc0330c8
online
```

Example 2 Describes the steps in taking an engine offline that is currently running tasks with an affinity for this engine:

```
select engine, status from sysengines

engine    status
------    ------
0         online
1         online
2         online
3         online
```

If you bring engine 1 offline:

```
sp_engine offline, 1
The following task(s) will affect the offline process:
spid: 19 has outstanding ct-lib connections.
```

And then run the same query as above, it now shows that engine 1 is in an offline state:

```
select engine, status from sysengines

engine    status
------    ------
0         online
1         in offline
2         online
3         online
```

As soon as the task that has an affinity to engine 1 finishes, Adaptive Server issues a message similar to the following to the error log:

```
02:00000:00000:2001/10/26 09:02:09.05 kernel  engine 1, os pid
8623  offline
```

Example 3 Determines whether engine 1 can be brought offline:

```
sp_engine can_offline, 1
```

Example 4 Takes engine 1 offline:

```
sp_engine offline, 1
```

Adaptive Server eventually returns a message similar to the following:

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting for
affinitated process(es) before going offline
01:00000:00000:2001/11/09 16:11:11.85 kernel  Process 917518 is preventing
engine 1 going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid
21127  offline
```

Example 5 Shuts down engine 1 :

```
sp_engine shutdown, 1
```

Usage

- You cannot offline or shut down engine 0.
- You can determine the status of an engine, and which engines are currently online with the following query:

  ```
  select engine, status from sysengines
  where status = "online"
  ```

- online and shutdown are keywords and must be enclosed in quotes.
- Engines can be brought online only if max online engines is greater then the current number of engines with an online status, and if enough CPU is available to support the additional engine.
- An engine offline may fail or may not immediately take effect if there are server processes with an affinity to that engine.

Using sp_engine “offline” versus sp_engine “shutdown”

Sometimes when you use sp_engine “offline", the engine does not immediately go offline, and instead appears to be in “dormant” state in the engine table. This is caused by processes that are attached to your engine that cannot be migrated to other engines. When this happens, the engine does not take new work, and consumes minimal CPU cycles. When the process preventing the completion of engine offline either end or become available for migration, the engine moves from dormant to fully offline, and disappears from the engine table.

sp_engine "shutdown" is a more aggressive version of the offline command. The sp_engine "shutdown" procedure actively kills any processes that are preventing the engine from going offline, forcing it to shut down.

However, if you use sp_engine "shutdown" on an engine that has ct-lib or java connections, you get the following error message:

```
Engine has outstanding ct-lib/java connections and
cannot be offlined.
```

When this happens, repeat the command again every few minutes until the connections are no longer there, and the engine can shut down.

Permissions

You must be a System Administrator to bring engines online or offline.

sp_estspace

Description

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*, *page_size*

Parameters

table_name
is the name of the table. It must already exist 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 that Adaptive Server uses its default fillfactor.

cols_to_max
is a comma-separated list of the variable-length columns for which you want to use the maximum length instead of the average. The default is the average declared length of the variable-length columns.

textbin_len
is the length, per row, of all text and image columns. The default value is 0. You need to provide a value only if the table stores text 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 or 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.

pagesize
allows you to estimate the space required for a given table—and all of its indexes—if you migrate the table to a server of the specified page size. You can either specify a page size (2048, 4096, 8192, 16384, or 2K, 4K, 8K, 16K) or NULL to use your current page size. If you do not use "K" as a unit specifier, the default for *pagesize* is bytes. Because page allocation allocates the same size page for various objects, the *page_size* value applies to all page types (index, data, text and so on).

Examples

Example 1 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 is 50 percent, two variable-length columns are computed using the maximum size for the column, and the disk I/O speed is 25 I/Os per second:

```
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
```

Example 2 Uses the average length of existing image data in the au_pix table to calculate the size of the table with 1000 rows. You can also provide this size as a constant:

```
declare @i int
select @i = avg(datalength(pic)) from au_pix
exec sp_estspace au_pix, 1000, null, null, 16, @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
```

Example 3 Calculates the size of the titles table with 50,000 rows, using defaults for all other values:

```
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
```

Example 4 This example is run after adding a clustered index to the blurbs table:

```
declare @i int
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 6, null, null, 16, @i, "16k"
```

```
name                     type        idx_level Pages         Kbytes
------------------------ ----------- --------- ---------     -------
blurbs                   data                0         8        128
blurbs                   text/image          0         6         96
blurbs_ind               clustered           0         1         16
blurbs_ind               clustered           1         1         16

Total_Mbytes
-----------------
0.25

name                     type         total_pages  time_mins
------------------------ ------------ ------------ ------------
blurbs_ind               clustered              10            0
blurbs                   data                    6            0
```

This example is run on a 2K server, and indicates that the blurbs table would require .25MB after it is migrated to a 16K server. Below is the same query run on a 16K server, which verifies the .25MB space requirement:

```
declare @i int
```

```
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 6, null, null, 16, @i, "16k"

name                     type         idx_level Pages     Kbytes
------------------------ ------------ --------- --------- ------
blurbs                   data                 0         8    128
blurbs                   text/image           0         6     96
blurbs_ind               clustered            0         1     16
blurbs_ind               clustered            1         1     16

Total_Mbytes
-----------------
0.25

name                    type         total_pages  time_mins
----------------------- ------------ ------------ ----------
blurbs_ind              clustered              10          0
blurbs                  data                    6          0
```

Example 5 This example estimates that, if the blurbs table had a thousand rows in it on a 2K server, it would require 1.99MB of space:

```
declare @i int
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 1000, null, null, 16, @i, "2k"

name                    type         idx_level Pages        Kbytes
----------------------- ------------ --------- ------------ ------
blurbs                  data                 0     16           32
blurbs                  text/image           0   1000         2000
blurbs_ind              clustered            0      1            2
blurbs_ind              clustered            1      1            2

Total_Mbytes
------------
1.99

name                    type         total_pages  time_mins
----------------------- ------------ ------------ ---------
blurbs_ind              clustered              18         0
blurbs                  data                 1000         0
```

Usage

- To estimate the amount of space required by a table and its indexes:
  a. Create the table.
  b. Create all indexes on the table.

c Run sp_estspace, giving the table name, the estimated number of rows for the table, 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 set in a database, Adaptive 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:

  a In the master database, use sp_dboption to turn on the auto identity option for the database.

  b Create the table.

  c Run sp_estspace on the table and record the results.

  d Drop the table.

  e Turn the auto identity option off for the database.

  f Re-create the table.

  g Rerun sp_estspace on the table, and record the results.

- For information about tables or columns, use sp_help *tablename*.

Permissions

Any user can execute sp_estspace.

See also

Commands create index, create table

System procedures sp_dboption, sp_help

sp_export_qpgroup

Description: Exports all plans for a specified user and abstract plan group to a user table.

Syntax: sp_export_qpgroup *usr*, *group*, *tab*

Parameters:

usr
is the name of the user who owns the abstract plans to be exported.

group
is the name of the abstract plan group that contains the plans to be exported.

tab
is the name of a table into which to copy the plans. It must be a table in the current database. You can specify a database name, but not an owner name, in the form *dbname..tablename*. The total length must be 30 characters or less.

Examples: Creates a table called moveplans containing all the plans for the user “freidak” that are in the ap_stdout group:

```
sp_export_qpgroup freidak, ap_stdout, "tempdb..moveplans"
```

Usage:

- sp_export_qpgroup copies plans from an abstract plan group to a user table. With sp_import_qpgroup, it can be used to copy abstract plans groups between servers and databases or to assign user IDs to copied plans.
- The user table name that you specify cannot exist before you run sp_export_qpgroup. The table is created with a structure identical to that of sysqueryplans.
- sp_export_qpgroup uses select...into to create the table to store the copied plans. You must use sp_dboption to enable select into/bulkcopy/pllsort in order to use sp_export_qpgroup, or create the table in tempdb.

Permissions: Only a System Administrator or the Database Owner can execute sp_export_qpgroup.

See also: **System procedures** sp_copy_all_qplans, sp_copy_qplan, sp_dboption, sp_import_qpgroup

sp_extendsegment

Description

Extends the range of a segment to another 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 be added to the current database device range already included in *segname*.

Examples

Extends the range of the segment indexes for the database pubs2 on the database device dev2:

```
sp_extendsegment indexes, pubs2, dev2
```

Usage

- A segment can be extended over several database devices.
- If the logsegment segment is extended, any other segments on the device are dropped and the device is used for the log segment exclusively.
- When you extend the logsegment segment, Adaptive Server recalculates its last-chance threshold.
- 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.
- After defining a segment, you can 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, subsequent data for the table or index is located on that segment.

Permissions

Only the Database Owner or a System Administrator can execute sp_extendsegment.

See also

Commands alter database, create index, create table

System procedures sp_addsegment, sp_dropsegment, sp_helpdb, sp_helpdevice, sp_helpsegment, sp_placeobject

sp_extengine

Description

Starts and stops EJB Server. Displays status information about EJB Server.

Syntax

sp_extengine '*ejb_server*', '{ start | stop | status }'

Parameters

ejb_server
the logical name of the EJB Server.

start
starts the EJB Server.

stop
shuts down the EJB Server.

status
displays status information about the EJB Server.

Examples

Example 1 Informs user that the EJB Server SYB_EJB is running:

```
sp_extengine 'SYB_EJB', 'status'
```

```
Enterprise java bean server is up and running.
```

Example 2 Shuts down the EJB Server SYB_EJB:

```
sp_extengine 'SYB_EJB', 'stop'
```

Usage

- You must have a valid Adaptive Server EJB Server site license to use sp_extengine.

Permissions

Only a System Administrator can execute sp_extengine.

See also

Documents See the *User's Guide to EJB Server* for more information.

sp_familylock

Description

Reports information about all the locks held by a family (coordinating process and its worker processes) executing a statement in parallel.

Syntax

sp_familylock [*fpid1* [, *fpid2*]]

Parameters

fpid1
is the family identifier for a family of worker processes from the master.dbo.sysprocesses table. Run sp_who or sp_lock to get the *spid* of the parent process.

fpid2
is the Adaptive Server process ID number for another lock.

Examples

Displays information about the locks held by all members of the family with an fid of 5:

```
sp_familylock 5

fid spid locktype    table_id page dbname class         context
--- ---- ----------   -------- ---- ------ ------------ --------
5   5    Sh_intent  176003658 0    userdb Non cursor lock Sync-pt duration
request
5   5    Sh_intent-blk 208003772 0    userdb Non cursor lock Sync-pt
duration request
5   6    Sh_page    208003772 3972 userdb Non cursor lock Sync-pt duration
request
5   7    Sh_page    208003772 3973 userdb Non cursor lock Sync-pt duration
request
5   8    Sh_page    208003772 3973 userdb Non cursor lock Sync-pt duration
request
```

Usage

- sp_familylock with no parameter reports information on all processes belonging to families that currently hold locks. The report is identical to the output from sp_lock; however, sp_familylock allows you to generate reports based on the family ID, rather than the process ID. It is useful for detecting family deadlocks.
- Use the object_name system function to derive a table's name from its ID number.
- The "locktype" column indicates whether the lock is a shared lock ("Sh" prefix), an exclusive lock ("Ex" prefix) or an update lock, and whether the lock is held on a table ("table" or "intent") or on a page ("page").

The “blk” suffix in the “locktype” column indicates that this process is blocking another process that needs to acquire a lock. As soon as this process completes, the other process(es) moves forward. The “demand” suffix indicates that the process is attempting to acquire an exclusive lock.

- The “class” column indicates whether a lock is associated with a cursor. It displays one of the following:
    - “Non cursor lock” indicates that the lock is not associated with a cursor.
    - “Cursor Id *number*” indicates that the lock is associated with the cursor ID number for that Adaptive Server process ID.
    - A cursor name indicates that the lock is associated with the cursor *cursor_name* that is owned by the current user executing sp_lock.
- The “fid” column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for “fid” are as follows:
    - A zero value indicates that the task represented by the spid is executed in serial. It is not participating in parallel execution.
    - A nonzero value indicates that the task (spid) holding the lock is a member of a family of processes (identified by “fid”) executing a statement in parallel. If the value is equal to the spid, it indicates that the task is the coordinating process in a family executing a query in parallel.
- The “context” column identifies the context of the lock. Worker processes in the same family have the same context value. Values for “context” are as follows:
    - “NULL” means that the task holding this lock is either executing a query in serial or is a query being executed in parallel in transaction isolation level 1.
    - “FAM_DUR” means that the task holding the lock will hold the lock until the query is complete.

      A lock’s context may be “FAM_DUR” if the lock is a table lock held as part of a parallel query, if the lock is held by a worker process at transaction isolation level 3, or if the lock is held by a worker process in a parallel query and must be held for the duration of the transaction.

Permissions

Any user can execute sp_familylock.

See also

Commands kill, select

System procedures sp_lock, sp_who

sp_find_qplan

Description Finds an abstract plan, given a pattern from the query text or plan text.

Syntax sp_find_qplan *pattern* [, group]

Parameters *pattern*
is a string to find in the text of the query or abstract plan.

group
is the name of the abstract plan group.

Examples **Example 1** Reports on all abstract plans that have the string “from titles” in the query:

```
sp_find_qplan "%from titles%"

gid id          text
--- ----------- --------------------------------------------------
2   921054317 select count(*) from titles
2   921054317
         ( plan
        ( i_scan t_pub_id_ix titles )
        ( )
)
( prop titles
        ( parallel 1 )
        ( prefetch 16 )
        ( lru )
)
5   937054374 select type, avg(price) from titles group by type
5   937054374
         ( plan
        ( store Worktab1
                ( i_scan type_price titles )
        )
        ( t_scan ( work_t Worktab1 ) )
)
( prop titles
        ( parallel 1 )
        ( prefetch 16 )
        ( lru )
```

Example 2 Finds all plans that include a table scan operator:

```
sp_find_qplan "%t_scan%"
```

Example 3 Uses the range pattern matching to look for strings such as “table1”, “table2”, and so forth, in plans in the dev_plans group:

```
sp_find_qplan "%table[0-9]%", dev_plans
```

Usage

- Use sp_find_qplan to find an abstract plan that contains a particular string. You can match strings from either the query text or from the abstract plan text.
- For each matching plan, sp_find_qplan prints the group ID, plan ID, query text and abstract plan text.
- If you include a group name, sp_find_qplan searches for the string in the specified group. If you do not provide a group name, sp_find_plan searches all queries and plans for all groups.
- You must supply the "%" wildcard characters, as shown in the examples, unless you are searching for a string at the start or end of a query or plan. You can use any Transact-SQL pattern matching syntax, such as that shown in Example 3.
- The text of queries in sysqueryplans is broken into 255-byte column values. sp_find_qplan may miss matches that span one of these boundaries, but finds all matches that are less than 127 bytes, even if they span two rows.

Permissions

Any user can execute sp_find_qplan. It reports only on abstract plans owned by the user who executes it, except when executed by a System Administrator or the Database Owner.

See also

System procedures sp_help_qpgroup, sp_help_qplan

sp_fixindex

Description

Repairs the index on one of your system tables when it has been corrupted.

Syntax

sp_fixindex *dbname*, *table_name*, *index_id*

Parameters

dbname
is the database name

table_name
is the table name

index_id
is the ID of the index you want to fix

Examples

In this example, sp_fixindex repairs the clustered index on the sysprocedures table of the pubs2 database:

```
1> sp_fixindex pubs2, sysprocedures, 1
2> go
```

Usage

Warning! Do not run sp_fixindex on the clustered index of the sysobjects or sysindexes tables or on user tables. If you do, sp_fixindex returns the following error message:

```
The index with id 1 on sysobjects cannot be recreated.
```

Before you run sp_fixindex, make sure your database is in single-user mode, and is reconfigured to allow updates to system tables.

After you run sp_fixindex:

- Use the dbcc checktable command to verify that the corrupted index has been fixed
- Disallow updates to system tables using sp_configure
- Turn off single-user mode

Do not run sp_fixindex on user tables.

Repairing a nonclustered index on sysobjects using sp_fixindex requires additional steps.

Permissions

Only SA can run sp_fixindex.

See also

Documents For more information on sp_fixindex, see:

- Chapter 2, “Encyclopedia of Tasks” in the *Troubleshooting and Error Message Guide*.

- Chapter 13, See “Indexing for Performance” in the *Performance and Tuning Guide: Basics*.

sp_flushstats

Description

Flushes statistics from in-memory storage to the systabstats system table.

Syntax

sp_flushstats *objname*

Parameters

objname
is the name of a table.

Examples

Flushes statistics for the titles table:

```
sp_flushstats titles
```

Usage

- Some statistics in the systabstats table are updated in in-memory storage locations and flushed to systabstats periodically, to reduce overhead and contention on systabstats.
- If you query systabstats using SQL, executing sp_flushstats guarantees that in-memory statistics are flushed to systabstats.
- The optdiag command always flushes in-memory statistics before displaying output.
- The statistics in sysstatistics are changed only by data definition language commands and do not require the use of sp_flushstats.

Permissions

Only a System Administrator can execute sp_flushstats.

sp_forceonline_db

Description

Provides access to all the pages in a database that were previously marked suspect by recovery.

Syntax

```
sp_forceonline_db dbname,
      {"sa_on" | "sa_off" | "all_users"}
```

Parameters

dbname
is the name of the database to be brought online.

sa_on
allows only users with the sa_role access to the specified page.

sa_off
revokes access privileges created by a previous invocation of sp_forceonline_page with sa_on.

all users
allows all users access to the specified page.

Examples

Example 1 Allows the System Administrator access to all suspect pages in the pubs2 database:

```
sp_forceonline_db pubs2, "sa_on"
```

Example 2 Revokes access to all suspect pages in the pubs2 database from the System Administrator. Now, no one can access the suspect pages in pubs2:

```
sp_forceonline_db pubs2, "sa_off"
```

Example 3 Allows all users access to all pages in the pubs2 database:

```
sp_forceonline_db pubs2, "all_users"
```

Usage

- A page that is forced online is not necessarily repaired. Corrupt pages can also be forced online. Adaptive Server does not perform any consistency checks on pages that are forced online.
- sp_forceonline_page with all users cannot be reversed. When pages have been brought online for all users, you cannot take them offline again.
- sp_forceonline_db cannot be used in a transaction.
- To bring only specific offline pages online, use sp_forceonline_page.

Permissions

Only a System Administrator can execute sp_forceonline_db.

See also

System procedures sp_forceonline_page, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_granularity, sp_setsuspect_threshold

sp_forceonline_object

Description

Provides access to an index previously marked suspect by recovery.

Syntax

sp_forceonline_object *dbname*, *objname*, *indid*,
    {sa_on | sa_off | all_users} [, no_print]

Parameters

dbname
is the name of the database containing the index to be brought online.

objname
is the name of the table.

indid
is the index ID of the suspect index being brought online.

sa_on
allows only users with the sa_role to access the specified index.

sa_off
revokes access privileges created by a previous invocation of sp_forceonline_object with sa_on.

all_users
allows all users to access the specified index.

no_print
skips printing a list of other suspect objects after the specified object is brought online.

Examples

Example 1 Allows a System Administrator to access the index with indid 3 on the titles table in the pubs2 database:

```
sp_forceonline_object pubs2, titles, 3 , sa_on
```

Example 2 Revokes access to the index from the System Administrator. Now, no one has access to this index:

```
sp_forceonline_object pubs2, titles, 3, sa_off
```

Example 3 Allows all users to access the index on the titles table in the pubs2 database:

```
sp_forceonline_object pubs2, titles, 3, all_users
```

Usage

- If an index on a data-only-locked table has suspect pages, the entire index is taken offline during recovery. Offline indexes are not considered by the query optimizer. Indexes on allpages-locked tables are not taken completely offline during recovery; only individual pages of these indexes are taken offline. These pages can be brought online with sp_forceonline_page.

- Use sp_listsuspect_object to see a list of databases that are offline.
- To repair a suspect index, use sp_forceonline_object with sa_on access. Then, drop and re-create the index.

Note If the index is on systabstats or sysstatistics (the only data-only-locked system tables) call Sybase Technical Support for assistance.

- sp_forceonline_object with all_users cannot be reversed. When an index has been brought online for all users, you cannot take it offline again.
- An index that is forced online is not necessarily repaired. Corrupt indexes can be forced online. Adaptive Server does not perform any consistency checks on indexes that are forced online.
- sp_forceonline_object cannot be used in a transaction.
- sp_forceonline_object works only for databases in which the recovery fault isolation mode is "page." Use sp_setsuspect_granularity to display the recovery fault isolation mode for a database.
- To bring all of a database's offline pages and indexes online in a single command, use sp_forceonline_db.

Permissions

Only a System Administrator can execute sp_forceonline_object.

See also

Documents For more information on recovery fault isolation, see the *System Administration Guide*.

System procedures sp_listsuspect_object, sp_setsuspect_granularity

sp_forceonline_page

Description

Provides access to pages previously marked suspect by recovery.

Syntax

sp_forceonline_page *dbname*, *pgid*,
    {"sa_on" | "sa_off" | "all_users"}

Parameters

dbname
is the name of the database containing the pages to be brought online.

pgid
is the page identifier of the page being brought online.

sa_on
allows only users with the sa_role access to the specified page.

sa_off
revokes access privileges created by a previous invocation of sp_forceonline_page with sa_on.

all_users
allows all users access to the specified page.

Examples

Example 1 Allows a System Administrator access to page 312 in the pubs2 database:

```
sp_forceonline_page pubs2, 312, "sa_on"
```

Example 2 Revokes access to page 312 in the pubs2 database from the System Administrator. Now, no one has access to this page:

```
sp_forceonline_page pubs2, 312, "sa_off"
```

Example 3 Allows all users access to page 312 in the pubs2 database:

```
sp_forceonline_page pubs2, 312, "all_users"
```

Usage

- sp_forceonline_page with all_users cannot be reversed. When pages have been brought online for all users, you cannot take them offline again.
- A page that is forced online is not necessarily repaired. Corrupt pages can also be forced online. Adaptive Server does not perform any consistency checks on pages that are forced online.
- sp_forceonline_page cannot be used in a transaction.
- sp_forceonline_page works only for databases in which the recovery fault isolation mode is "page." Use sp_setsuspect_granularity to display the recovery fault isolation mode for a database.
- To bring all of a database's offline pages online in a single command, use sp_forceonline_db.

Permissions	Only a System Administrator can use sp_forceonline_page.
See also	**System procedures** sp_forceonline_db, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_granularity, sp_setsuspect_threshold

sp_foreignkey

Description

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 to be defined.

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 have at least one column and can have a maximum of eight columns.

Examples

Example 1 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:

```
sp_foreignkey titles, publishers, pub_id
```

Example 2 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:

```
sp_foreignkey orders, parts, part, subpart
```

Usage

- sp_foreignkey adds the key to the syskeys table. Keys make explicit a logical relationship that is implicit in your database design.
- sp_foreignkey does not enforce referential integrity constraints; use the foreign key clause of the create table or alter table command to enforce a foreign key relationship.
- 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 lengths) of the primary and foreign keys must agree, but the null types need not agree.
- The installation process runs sp_foreignkey on the appropriate columns of the system tables.
- To display a report on the keys that have been defined, execute sp_helpkey.
- You cannot use a Java datatype with sp_foreignkey.

Permissions

Only the owner of the table or view can execute sp_foreignkey.

See also

Commands alter table, create table, create trigger

System procedures sp_commonkey, sp_dropkey, sp_helpjoins, sp_helpkey, sp_primarykey

sp_freedll

Description

Unloads a dynamic link library (DLL) that was previously loaded into XP Server memory to support the execution of an extended stored procedure (ESP).

Syntax

sp_freedll *dll_name*

Parameters

dll_name
is the file name of the DLL being unloaded from XP Server memory.

Examples

Unloads the *sqlsrvdll.dll* DLL:

```
sp_freedll "sqlsrvdll.dll"
```

Usage

- sp_freedll cannot be executed from within a transaction.
- sp_freedll cannot free the DLL of a system ESP.
- An alternative to unloading a DLL explicitly, using sp_freedll, is to specify that DLLs always be unloaded after the ESP request that invoked them terminates. To do this, set the esp unload dll configuration parameter to 1 or start xpserver with the -u option.
- sp_freedll can be used to update an ESP function in a DLL without shutting down XP Server or Adaptive Server.
- If you use sp_freedll to unload a DLL that is in use, sp_freedll will succeed, causing the ESP currently using the DLL to fail.

Permissions

Only a System Administrator can execute sp_freedll.

See also

System procedures sp_addextendedproc, sp_dropextendedproc, sp_helpextendedproc

sp_getmessage

Description

Retrieves stored message strings from sysmessages and sysusermessages for print and raiserror statements.

Syntax

sp_getmessage *message_num*, *result* output [, *language*]

Parameters

message_num
is the number of the message to be retrieved.

result 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, unichar, nchar, varchar, univarchar, or nvarchar.

language
is the language of the message to be retrieved. *language* 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 you do not include *language*, then the message for the default session language, as indicated by the variable *@@langid*, is retrieved.

Examples

Example 1 Retrieves message number 20001 from sysusermessages:

```
declare @myvar varchar(200)
exec sp_getmessage 20001, @myvar output
```

Example 2 Retrieves the French language version of message number 20010 from sysusermessages:

```
declare @myvar varchar(200)
exec sp_getmessage 20010, @myvar output, french
```

Usage

- Any application can use sp_getmessage, and any user can read the messages stored in sysmessages and sysusermessages.

Permissions

Any user can execute sp_getmessage.

See also

Commands print, raiserror

System procedures sp_addmessage, sp_dropmessage

sp_grantlogin

Description

Windows NT only Assigns Adaptive Server roles or default permissions to Windows NT users and groups when Integrated Security mode or Mixed mode (with Named Pipes) is active.

Syntax

sp_grantlogin {*login_name* | *group_name*}
    ["*role_list*" | default]

Parameters

login_name
is the network login name of the Windows NT user.

group_name
is the Windows NT group name.

role_list
is a list of the Adaptive Server roles granted. The role list can include one or more of the following role names: sa_role, sso_role, oper_role. If you specify more than one role, separate the role names with spaces, not commas.

default
specifies that the *login_name* or *group_name* receive default permissions assigned with the grant statement or sp_role procedure.

Examples

Example 1 Assigns the Adaptive Server oper_role to the Windows NT user "jeanluc":

```
sp_grantlogin jeanluc, oper_role
```

Example 2 Assigns the default value to the Windows NT user "valle". User "valle" receives any permissions that were assigned to her via the grant command or sp_role procedure:

```
sp_grantlogin valle
```

Example 3 Assigns the Adaptive Server sa_role and sso_role to all members of the Windows NT administrators group:

```
sp_grantlogin Administrators, "sa_role sso_role"
```

Usage

- You must create the Windows NT login name or group before assigning roles with sp_grantlogin. See your Windows NT documentation for details.
- sp_grantlogin is active only when Adaptive Server is running in Integrated Security mode or Mixed mode when the connection is Named Pipes. If Adaptive Server is running under Standard mode or Mixed mode with a connection other than Named Pipes, use grant and sp_role instead.
- If you do not specify a *role_list* or default, the procedure automatically assigns the default value.

- The default value does not indicate an Adaptive Server role. It specifies that the user or group should receive any permissions that were assigned to it via the grant command or sp_role procedure.
- Using sp_grantlogin with an existing *login_name* or *group_name* overwrites the user's or group's existing roles.

Permissions

Only a System Administrator can execute sp_grantlogin.

See also

Commands grant, setuser

System procedures sp_addlogin, sp_displaylogin, sp_droplogin, sp_locklogin, sp_logininfo, sp_modifylogin, sp_revokelogin, sp_role

sp_ha_admin

Description

Performs administrative tasks on Adaptive Servers configured with Sybase Failover in a high availability system. sp_ha_admin is installed with the *installhavss* script on UNIX platforms or the *insthasv* script on Windows NT.

Syntax

sp_ha_admnin [cleansessions | help]

Parameters

cleansessions
removes old entries from syssessions. Old syssessions entries are typically left behind because either Adaptive Server failed to clean up syssessions during a reboot, or because a client failed to connect to Adaptive Server.

help
displays the syntax for sp_ha_admin.

Examples

Example 1 Removes old entries from syssessions left by a client connection that did not exit correctly:

```
sp_ha_admin cleansessions
(return status = 0)
```

Example 2 Displays the syntax for sp_ha_admin:

```
sp_ha_admin "help"

sp_ha_admin Usage: sp_ha_admin command [, option1 [,
option2]]
sp_ha_admin commands:
sp_ha_admin 'cleansessions'
sp_ha_admin 'help'
(return status = 0)
```

Usage

- sp_ha_admin performs administrative tasks on Adaptive Server that are configured for Sybase's Failover in a high availability system. sp_ha_admin is not installed using the *installmaster* script; instead, use the *installhavss* script that installs and configures for Sybase's Failover (*insthasv* on Windows NT).
- sp_ha_admin returns a 0 if it successfully cleaned up syssessions, and returns a 1 if it encounters an error.
- sp_ha_admin enters a message in the errorlog if it could not remove any entries from syssessions (for example, if it could not get a lock on syssessions).
- To view all the current entries in syssessions, enter:

```
select * from syssessions
```

Permissions

Only the a System Administrator with the ha_role can execute sp_ha_admin.

sp_help

Description

Reports information about a database object (any object listed in sysobjects) and about system or user-defined datatypes. Column displays optimistic_index_lock.

Syntax

sp_help [*objname*]

Parameters

objname

is the name of any object in sysobjects or 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, but refers to tables when you enable optimistic_index_lock. Use owner names if the object owner is not the user running the command and is not the Database Owner.

Examples

Example 1 Displays a list of objects in sysobjects and displays each object's name, owner, and object type. Also displays a list of each user-defined datatype in systypes, indicating the datatype name, storage type, length, null type, default name, and rule name. Null type is 0 (null values not allowed) or 1 (null values allowed):

```
sp_help
```

Example 2 Displays information about the publishers table. sp_help also lists any attributes assigned to the specified table and its indexes, giving the attribute's class, name, integer value, character value, and comments. The above example shows cache binding attributes for the publishers table:

```
sp_help publishers

Name                       Owner                          Object_Type
-------------------------- ---------------------------    -----------
publishers                 dbo                            user table
(1 row affected)

 Data_located_on_segment         When_created
------------------------------       --------------------------
default                                              Apr 25 2002 10:28AM

Column_name Type   Length   Prec   Scale    Nulls  Default_name
Rule_name        Access_Rule_name              Identity
---------- ----  ------ -----  -----  ----- -----------  -----------
--------------- ------------------------------ --------
pub_id     char        4    NULL    NULL      0           NULL
pub_idrule          NULL                              0
pub_name   varchar   40    NULL    NULL      1           NULL
NULL                NULL                              0
city       varchar   20    NULL    NULL      1           NULL
```

```
NULL              NULL                                0
state       char       2    NULL    NULL      1         NULL
NULL              NULL                                0

index_name      index_description                      index_keys


index_max_rows_per_page    index_fillfactor    index_reservepagegap
index_created
-----------     ----------------------------------      -----------------
-------------------------     --------------------   --------------------
pubind          clustered, unique located on default     pub_id

0                               0                        0
Apr 25 2002 10:28AM
(1 row affected)

keytype        object                 related_object
object_keys                     related_keys
----------    ------------           ------------------
---------------------           ---------------------
primary       publishers             -- none --
pub_id, *, *, *, *, *, *, *     *, *, *, *, *, *, *, *
foreign       titles                 publishers
pub_id, *, *, *, *, *, *,        pub_id, *, *, *, *, *, *, *
(1 row affected)
Object is not partitioned.
Lock scheme Allpages
The attribute 'exp_row_size' is not applicable to tables with allpages lock
scheme.
The attribute 'concurrency_opt_threshold' is not applicable to tables with
allpages lock scheme.
exp_row_size reservepagegap fillfactor max_rows_per_page identity_gap
------------ -------------- ---------- ----------------- ------------
           0              0          0                 0            0
concurrency_opt_threshold
-------------------------
                           0
```

Example 3 Displays information about a partitioned table (in this example, the titles table was first altered to have four partitions):

```
sp_help titles

Name                       Owner                              Object_Type
-------------------------- ---------------------------      -----------
titles                     dbo                                user table
(1 row affected)
```

```
 Data_located_on_segment         When_created
------------------------------       --------------------------
default                                           Apr 25 2002 10:28AM

Column_name Type Length   Prec  Scale  Nulls Default_name
Rule_name        Access_Rule_name              Identity
---------- ----  ------- -----  -----  ----- ------------
--------------- ------------------------------ --------
title_id    tid         6  NULL   NULL      0         NULL
title_idrule                              NULL        0
title   varchar       80  NULL   NULL      0         NULL
NULL                                      NULL        0
type       char       12  NULL   NULL      0     typedflt
NULL                                      NULL        0
pub_id     char        4  NULL   NULL      1         NULL
NULL                                      NULL        0
price     money        8  NULL   NULL      1         NULL
NULL                                      NULL        0
advance   money        8  NULL   NULL      1         NULL
NULL                                      NULL        0
total_sales int        4  NULL   NULL      1         NULL
NULL                                      NULL        0
notes   varchar      200  NULL   NULL      1         NULL
NULL                                      NULL        0
pubdate datetime       8  NULL   NULL      0     datedflt
NULL                                      NULL        0
contract    bit        1  NULL   NULL      0         NULL
NULL                                      NULL        0
index_name      index_description                       index_keys


index_max_rows_per_page    index_fillfactor    index_reservepagegap
index_created
-----------     ----------------------------------      -----------------
-----------------------    -------------------- -------------------
--------------
titleidind        clustered, unique located on default          title_id
Apr 25 2002 10:28AM
titleind          nonclustered located on default               title
Apr 25 2002 10:28AM

(2 rows affected)

keytype        object                related_object
object_keys                     related_keys
----------    ------------          ------------------
---------------------           ---------------------
```

```
foreign          roysched              titles
title_id, *, *, *, *, *, *, *   title_id, *, *, *, *, *, *, *
foreign          salesdetail           titles
title_id, *, *, *, *, *, *, *   title_id, *, *, *, *, *, *,
foreign          titleauthor           titles
title_id, *, *, *, *, *, *, *   title_id, *, *, *, *, *, *,
foreign          titles                publishers
pub_id, *, *, *, *, *, *, *     pub_id, *, *, *, *, *, *, *
primary          titles                -- none --
title_id, *, *, *, *, *, *, *   *, *, *, *, *, *, *, *

(1 row affected)
partitionid       firstpage      controlpage       ptn_data_pages
-----------       -----------    ----------        --------------
          1              784            785                     1
          2              713            712                     1
          3              721            720                     1
          4              945            944                     1
Partitions  Average Pages  Maximum Pages  Minimum Pages     Ratio (Max/Avg)
----------    -------------    --------------    -------    ----------
         4              1                 1          1      1.000000
Lock scheme Allpages
The attribute 'exp_row_size' is not applicable to tables with allpages lock
scheme.
The attribute 'concurrency_opt_threshold' is not applicable to tables with
allpages lock scheme.
exp_row_size reservepagegap fillfactor max_rows_per_page identity_gap
------------ -------------- ---------- ----------------- ------------
           0              0          0                 0            0
concurrency_opt_threshold
-------------------------
                           0
```

Example 4 Displays information about the trigger marytrig owned by user "mary". The quotes are needed, because the period is a special character:

```
sp_help "mary.marytrig"
```

```
Name         Owner              Object_type
------------ ------------------ ----------------
marytrig     mary              trigger

Data_located_on_segment When_created
----------------------- --------------------------
not applicable          Mar 20 2002  2:03PM
```

Example 5 Displays information about the system datatype money:

```
sp_help money

Type_name  Storage_type  Length  Prec   Scale  Nulls  Defaul_name
---------  ------------  -------  -----  -----  -----  ------------
Rule_name  Access_Rule_name  Identity
---------  ----------------  --------
money       money              8   NULL    NULL      1         NULL
NULL                   NULL         0
```

Example 6 Displays information about the user-defined datatype identype. The report indicates the base type from which the datatype was created, whether it allows nulls, the names of any rules and defaults bound to the datatype, and whether it has the IDENTITY property:

```
sp_help identype

Type_name  Storage_type  Length  Prec   Scale  Nulls  Defaul_name
---------  ------------  -------  -----  -----  -----  ------------
Rule_name  Access_Rule_name  Identity
---------  ----------------  --------
identype        numeric        4   NULL    NULL      1         NULL
NULL                   NULL         1
```

Shows a new column, indicating whether optimistic index locking is enabled. 1 indicates that the option is enabled; 0 indicates that it is not.

```
sp_help "mytable"
------------
exp_row_sizereservepagegapfillfactormax_rows_per_page
------------------------------------------------------
   1        0         0      0         0
concurrency_opt_thresholdoptimistic_index_lock
-------------------------------------------------
          0                         1
```

Usage

- sp_help looks for an object in the current database only.
- sp_help follows the Adaptive Server rules for finding objects:
  - If you do not specify an owner name, and you own an object with the specified name, sp_help reports on that object.
  - If you do not specify an owner name, and do not own an object of that name, but the Database Owner does, sp_help reports on the Database Owner's object.

    - If neither you nor the Database Owner owns an object with the specified name, sp_help reports an error condition, even if an object with that name exists in the database for a different owner. Qualify objects that are owned by database users other than yourself and the Database Owner with the owner's name, as shown in Example 4.
    - If both you and the Database Owner own objects with the specified name, and you want to access the Database Owner's object, specify the name in the format *dbo.objectname*.
- sp_help works on temporary tables if you issue it from tempdb.
- Columns with the IDENTITY property have an "Identity" value of 1; others have an "Identity" value of 0. In example 2, there are no IDENTITY columns.
- 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. It also lists any attributes associated with those indexes. However, sp_help does not describe any information about the integrity constraints defined for a table. Use sp_helpconstraint for information about any integrity constraints.
- sp_help displays the following new settings:
    - The locking scheme, which can be set with create table and changed with alter table
    - The expected row size, which can be set with create table and changed with sp_chgattribute
    - The reserve page gap, which can be set with create table and changed with sp_chgattribute
    - The row lock promotion settings, which can be set or changed with sp_setpglockpromote and dropped with sp_droprowlockpromote
- sp_help includes the report from sp_helpindex, which shows the order of the keys used to create the index and the space management properties.
- When Component Integration Services is enabled, sp_help displays information on the storage location of remote objects.

Permissions

Any user can execute sp_help.

See also

Documents sp_help reports information about SQLJ stored procedures and SQLJ functions. See *Java in Adaptive Server Enterprise* for more information about SQLJ routines.

Commands alter table, create table

System procedures sp_chgattribute, sp_droprowlockpromote, sp_helpconstraint, sp_helpindex, sp_setpglockpromote

sp_helpartition

Description

Lists the partition number, first page, control page, and number of data pages and summary size information for each partition in a partitioned table.

Syntax

sp_helpartition [*table_name*]

Parameters

table_name

is the name of a partitioned table in the current database. If the table name is not supplied, the owner, tables name, and number of partitions is printed for all user tables in the database.

Examples

Returns information about the partitions in sales:

```
sp_helpartition sales

partitionid firstpage   controlpage ptn_data_pages
----------- ----------- ----------- --------------
          1         313         314           4227
          2       12802       12801           4285
          3       25602       25601           4404
          4       38402       38401           4523
          5       51202       51201           4347
          6       64002       64001           4285

(6 rows affected)
Partitions Average Pages Maximum Pages Minimum Pages Ratio (Max/Avg)
---------- ------------- ------------- ------------- ---------------
         6          4345          4523          4227        1.040967
```

Usage

- sp_helpartition lists the partition number, first page, control page, and number of data pages for each partition in a partitioned table. The number of pages per partition shows how evenly the data is distributed between partitions.

  The summary information display the number of partitions, the average number of pages per partition, the minimum and maximum number of pages, and the ratio between the average number of pages and the maximum number. This ratio is used during query optimization. If the ratio is 2 or greater (meaning that the maximum size is twice as large as the average size), the optimizer chooses a serial query plan rather than a parallel plan.

- Partitioning a table creates additional page chains. Use the partition clause of the alter table command to partition a table. Each chain has its own last page, which is available for concurrent insert operations. This improves insert performance by reducing page contention. If the table is spread over multiple physical devices, partitioning improves insert performance by reducing I/O contention while Adaptive Server is flushing data from cache to disk.
- Partitioning a table does not affect its performance for update or delete commands.
- Use the unpartition clause of the alter table command to concatenate all existing page chains.
- Neither partitioning nor unpartitioning a table moves existing data.
- To change the number of partitions in a table, first use the unpartition clause of alter table to concatenate its page chains. Then use the partition clause of alter table to repartition the table.
- sp_helpartition looks only in the current database for the table.
- Use sp_helpsegment to display the number of used and free pages on the segment on where the partitioned table is stored.

Accuracy of results

- The values reported in the "data_pages" column may be greater than the actual values. To determine whether the count is inaccurate, run sp_statistics and sp_helpartition to compare the data page count. The count provided by sp_statistics is always accurate.

  If the page count reported by sp_statistics differs from the sum of the partition pages reported by sp_helpartition by more then 5 percent, run one of the following commands to update the partition statistics:

  - dbcc checkalloc
  - dbcc checkdb
  - dbcc checktable
  - update all statistics
  - update partition statistics

  Then, rerun sp_helpartition for an accurate report.

Permissions

Any user can execute sp_helpartition.

See also

Catalog system procedures sp_statistics

Commands alter table, insert

System procedures sp_helpsegment

sp_helpcache

Description

Displays information about the objects that are bound to a data cache or the amount of overhead required for a specified cache size.

Syntax

sp_helpcache {*cache_name* | "*cache_size*[P|K|M|G]"}

Parameters

cache_name
is the name of an existing data cache.

cache_size
specifies the size of the cache, specified by P for pages, K for kilobytes, M for megabytes, or G for gigabytes. The default is K.

Examples

Example 1 Displays information about items bound to pub_cache:

```
sp_helpcache pub_cache
```

Example 2 Shows the amount of overhead required to create an 80MB data cache:

```
sp_helpcache "80M"
```

Example 3 Displays information about all caches and all items bound to them:

```
sp_helpcache
```

Usage

- To see the size, status, and I/O size of all data caches on the server, use sp_cacheconfig.
- When you configure data caches with sp_cacheconfig, all the memory that you specify is made available to the data cache. Overhead for managing the cache is taken from the default data cache. The sp_helpcache displays the amount of memory required for a cache of the specified size.
- To bind objects to a cache, use sp_bindcache. To unbind a specific object from a cache, use sp_unbindcache. To unbind all objects that are bound to a specific cache, use sp_unbindcache_all.
- The procedure sp_cacheconfig configures data caches. The procedure sp_poolconfig configures memory pools within data caches.
- sp_helpcache computes overhead accurately up to 74GB.
- Although you can still use sp_bindcache on a system tempdb, the binding of the system tempdb is now non-dynamic. Until you restart the server, the changes do not take effect, and sp_helpcache reports a status of “P” for pending, unless you have explicitly bound the system tempdb to the default data cache, in which case the status as “V” for valid, because by default the system tempdb is already bound to the default datacache.

Permissions	Any user can execute sp_helpcache.
See also	**System procedures** sp_bindcache, sp_cacheconfig, sp_poolconfig, sp_unbindcache, sp_unbindcache_all

sp_helpconfig

Description

Reports help information on configuration parameters.

Syntax

sp_helpconfig "*configname*", ["*size*"]

Parameters

configname

is the configuration parameter being queried, or a non-unique parameter fragment.

size

is the size of memory, specified by B (bytes), K (kilobytes), M (megabytes), G (gigabytes), or P (pages). Used without the type of size specified, *size* specifies the number of the entity being configured using this parameter, for examples, locks, open indexes, and so on. *size* is ignored if *configname* is not a unique parameter name.

Examples

Example 1 Returns a report on all configuration options that start with "allow":

```
sp_helpconfig "allow"
```

```
Configuration option is not unique.
 option_name                    config_value run_value
 ------------------------------ ------------ -----------
 allow backward scans                      1           1
 allow nested triggers                     1           1
 allow procedure grouping                  1           1
 allow remote access                       1           1
 allow resource limits                     0           0
 allow sendmsg                             0           0
 allow sql server async i/o                1           1
 allow updates to system tables            0           0
```

Example 2 Returns a report on how much memory is needed to create a metadata cache for 421 object descriptors:

```
sp_helpconfig "open objects", "421"
```

```
number of open objects sets the maximum number of database objects that are
open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          243

Configuration parameter, 'number of open objects', will consume 207K of
memory if configured at 421.
```

Example 3 Returns a report on how many database descriptors would fill a 1MB database cache:

```
sp_helpconfig "open databases", "1M"
```

```
number of open databases sets the maximum number of databases that can be
open at one time on SQL Server. The default run value is 12.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
            5     2147483647             12             12          433

Configuration parameter, 'number of open databases', can be configured to
28 to fit in 1M of memory.
```

Example 4 Returns a report on how many locks will use 512K of memory:

```
sp_helpconfig "number of locks", "512K"
```

```
number of locks sets the number of available locks. The default run value
is 5000.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
         1000    2147483647           5000           5000           528

Configuration parameter 'number of locks', can be configured to 4848 to fit
in 512K of memory.
```

Example 5 Returns a report on the status of the allow updates to system tables configuration parameter:

```
sp_helpconfig "allow updates to system tables"
```

```
allow updates to system tables allows system tables to be updated directly.
The default is 0 (off).

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
            0              1              0              0            0
```

Usage

- sp_helpconfig reports help information on configuration parameters, such as how much memory would be needed if the parameter were set to a certain value. sp_helpconfig also displays the current setting, the amount of memory used for that setting, the default value, and the minimum and maximum settings.

Note The "maximum value" setting refers to the largest number that the parameter's datatype can accept, rather than to an actual configurable value.

In many cases, the maximum allowable values for configuration parameters are extremely high. The maximum value for your server is usually limited by available memory and other resources, rather than by configuration parameter limitations.

- If you use a nonunique parameter fragment for *configname*, sp_helpconfig returns a list of matching parameters with their configured values and current values. See Example 1.

Planning metadata cache configuration

- Use sp_helpconfig when you are planning a metadata cache configuration for a server.

  For example, suppose you were planning to move a database that contained 2000 user indexes to a different server. To find how much memory you would need to configure for that server so that it would accommodate the database's user indexes, enter the following command:

```
sp_helpconfig "open indexes", "2000"

number of open indexes sets the maximum number of indexes that can be
open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          208

Configuration parameter, 'number of open indexes', will consume 829k of
memory if configured at 2000.
```

  Alternatively, suppose you had 1MB of memory available for the index cache, and you needed to know how many index descriptors it would support. Run the following command:

```
sp_helpconfig "open indexes", "1M"

number of open indexes sets the maximum number of indexes that can be
```

```
open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          208

Configuration parameter 'number of open indexes', can be configured to
2461 to fit in 1M of memory.
```

Based on this output, if you have 1MB of memory, you can create an index descriptor cache that can contain a maximum of 2461 index descriptors. To create this cache, set the number of open indexes configuration parameter as follows:

```
sp_configure "number of open indexes", 2461
```

Using *sp_helpconfig* with *sybdiagdb* (Sybase Technical Support only)

Note Sybase Technical Support may create the sybdiagdb database on your system for debugging purposes. This database holds diagnostic configuration data, and is for use by Sybase Technical Support only.

The following *configname* options have been added to sp_helpconfig for Sybase Technical Support to use with the sybdiagdb database:

- *number of ccbs* – the number of configurable action point control blocks available to aid debugging.
- *caps per ccb* – the maximum number of configurable action points that can be configured at any one time within one configurable action point.
- *average cap size* – the estimated number of bytes of memory required to store the information associated with a typical configurable action point.

For example:

```
sp_helpconfig "number of ccbs"

Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
0             100             0             0            0


sp_helpconfig "caps per ccb"

Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
5             500            50            50            0
```

```
sp_helpconfig "average cap size"

Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
100         10000           200           200            0
```

Permissions The options specified in "Using sp_helpconfig with sybdiagdb (Sybase Technical Support only)" on page 272 can be used only by Sybase Technical Support. Any user can execute sp_helpconfig with other *configname* options.

See also **System procedures** sp_configure, sp_countmetadata, sp_monitorconfig

sp_helpconstraint

Description

Reports information about integrity constraints used in the specified tables.

Syntax

sp_helpconstraint [*objname*] [, detail]

Parameters

objname

is the name of a table that 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

Example 1 Displays the constraint information for the store_employees table in the pubs3 database. The store_employees table has a foreign key to the stores table (stor_id) and a self-reference (mgr_id references emp_id):

```
sp_helpconstraint store_employees

name                         defn
---------------------------  --------------------------------
store_empl_stor_i_272004000  store_employees FOREIGN KEY
                              (stor_id) REFERENCES stores(stor_id)
store_empl_mgr_id_288004057  store_employees FOREIGN KEY
                             (mgr_id) SELF REFERENCES
                             store_employees(emp_id)
store_empl_2560039432        UNIQUE INDEX( emp_id) :
                             NONCLUSTERED, FOREIGN REFERENCE

(3 rows affected)

Total Number of Referential Constraints: 2

Details:
-- Number of references made by this table: 2
-- Number of references to this table: 1
-- Number of self references to this table: 1

Formula for Calculation:
Total Number of Referential Constraints
= Number of references made by this table
+ Number of references made to this table
- Number of self references within this table
```

Example 2 Displays more detailed information about the pubs3..salesdetail constraints, including the constraint type and any constraint error messages:

```
sp_helpconstraint titles, detail

name                                type
```

```
      defn
        msg
------------------------------  ------------------------
     ------------------------------------------------------------
        --------------------------------------------
datedflt                        default value
      create default datedflt as getdate()

typedflt                        default value
      create default typedflt as "UNDECIDED"

titles_pub_id_96003373          referential constraint
      titles FOREIGN KEY (pub_id) REFERENCES publishers(pub_id)
        standard system error message number : 547

roysched_title__144003544       referential constraint
      roysched FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

salesdetai_title__368004342     referential constraint
      salesdetail FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

titleautho_title__432004570     referential constraint
      titleauthor FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

titles_800033162                unique constraint
      UNIQUE INDEX ( title_id) : NONCLUSTERED, FOREIGN REFERENCE
        standard system error message number : 2601

(7 rows affected)

Total Number of Referential Constraints: 4

Details:
-- Number of references made by this table: 1
-- Number of references to this table: 3
-- Number of self references to this table: 0

Formula for Calculation:
Total Number of Referential Constraints
= Number of references made by this table
+ Number of references made to this table
- Number of self references within this table.
```

Example 3 Displays a listing of all tables in the pubs3 database:

```
sp_helpconstraint

id          name                    Num_referential_constraints
----------- ----------------------- ---------------------------
   80003316 titles                                            4
   16003088 authors                                           3
  176003658 stores                                            3
  256003943 salesdetail                                       3
  208003772 sales                                             2
  336004228 titleauthor                                       2
  896006223 store_employees                                   2
   48003202 publishers                                        1
  128003487 roysched                                          1
  400004456 discounts                                         1
  448004627 au_pix                                            1
  496004798 blurbs                                            1

(11 rows affected)
```

Usage

- sp_helpconstraint prints the name and definition of the integrity constraint, and the number of references used by the table. The detail option returns information about the constraint's user or error messages.
- Running sp_helpconstraint with no parameters lists all the tables containing references in the current database, and displays the total number of references in each table. sp_helpconstraint lists the tables in descending order, based on the number of references in each table.
- 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, Adaptive Server reports the system error message associated with the constraint. Query sysmessages to obtain the actual text of that error message.
- You can use sp_helpconstraint only for tables in the current database.
- If a query exceeds the configured number of auxiliary scan descriptors, Adaptive Server returns an error message. You can use sp_helpconstraint to determine the necessary number of scan descriptors. See the *System Administration Guide* or more information on the number of aux scan descriptors configuration parameter.

- A System Security Officer can prevent the source text of constraint definitions from being displayed to most users who execute sp_helpconstraint. To restrict select permission on the text column of the syscomments table to the object owner or a System Administrator, use sp_configure to set the select on syscomments.text column parameter to 0. This restriction is required to run Adaptive Server in the evaluated configuration. See the *System Administration Guide* for more information about the evaluated configuration.

Permissions

Any user can execute sp_helpconstraint.

See also

Commands alter table, create table

System procedures sp_configure, sp_help, sp_helpdb, sp_monitorconfig

sp_helpdb

Description

Reports information about a particular database or about all databases. Displays a column for Asynchronous Log Service (ALS).

Syntax

sp_helpdb [*dbname*]

Parameters

dbname

is the name of the database on which to report information. Without this optional parameter, sp_helpdb reports on all databases. *dbname* can include wildcard characters to return all databases that match the specified pattern.

Examples

Example 1 Displays information about all the databases in Adaptive Server:

```
sp_helpdb

name            db_size  owner dbid created       status
--------------  -------- ----- ---- ------------- ------------------
master           5.0 MB   sa    1   Jan 01, 1900   no options set
model            2.0 MB   sa    3   Jan 01, 1900   no options set
pubs2            2.0 MB   sa    6   Sep 20, 1995   no options set
sybsystemprocs   16.0 MB  sa    4   Sep 20, 1995   trunc log on chkp
tempdb           2.0 MB   sa    2   Sep 20, 1995   select into/bulkcopy
```

Example 2 Issued from within pubs2, displays information about the pubs2 database, and includes segment information:

```
sp_helpdb pubs2

name   db_size  owner  dbid  created       status
-----  -------  -----  ----  ------------  ----------------------
pubs2  2.0 MB   sa     4     Mar 05, 1993  abort tran when log full
device_fragments   size    usage        free kbytes
-----------------  ------  ------------  -------------
master             2.0 MB  data and log           576
device                        segment
----------------------------- -----------------------------
master                        default
master                        logsegment
master                        system
name    attribute_class attribute     int_value char_value  comments
------- --------------- ------------- --------- ----------- --------
pubs2   buffer manager  cache binding         1 pubs2_cache     NULL
```

Example 3 Not issued from within pubs2, displays information about the pubs2 database:

```
sp_helpdb pubs2
```

```
name   db_size  owner  dbid  created       status

-----  -------  -----  ----  ------------  ---------------------
pubs2  2.0 MB    sa     4    Mar 05, 1993  abort tran when log full
device_fragments      size    usage           free kbytes
-----------------     ------  ------------  -------------
master                2.0 MB  data and log            576
name    attribute_class attribute     int_value char_value  comments
------- --------------- ------------- --------- ---------- --------
pubs2   buffer manager  cache binding         1 pubs2_cache     NULL
```

Example 4 Displays the row lock promotion attributes set for the pubtune database:

```
sp_helpdb pubtune
```

```
name                           attribute_class
attribute                      int_value
         char_value
         comments
-----------------------------------------------------------
pubtune                        lock strategy
     row lock promotion                    NULL
     PCT = 95, LWM = 300, HWM = 300
```

Example 5 Shows the advanced log service column in the output for sp_helpdb.

```
sp_helpdb "mydb"
```

```
------------
name   db_size   owner  dbid   created         status
--------------------------------------------------
mydb   3.0MB     sa     2      June 09, 2002   trunc log on chkpt
attribute
---------
async log serv
```

See Chapter 2, "Advanced Optmizing Tools" in *Performance and Tuning: Optimizer* guide for more information about advanced log service.

Example 6 Displays whether or not a database is a user-created temporary database under the status column:

```
sp_helpdb "mytempdb3"
```

```
name     db_size owner dbid created       status
-------  ------- ----- ---- -------       -----
mytempdb 32.0 MB sa    7    Dec 12, 2001   select into/bulkcopy/pllsort,
trunc
                                                   log on chkpt, user created temp
db
```

Usage

- sp_helpdb reports on the specified database when *dbname* is given. If no value is supplied for *dbname*, sp_helpdb reports on all the databases listed in master.dbo.sysdatabases.
- For log segment disk pieces in a dedicated log database, sp_helpdb issues "not applicable" for the free space field in its per-disk-piece repor. sp_helpdb also includes a column titled *free pages*, which is the value for the number of free pages the log segment has.
- *dbname* can include wildcard characters to return all databases that match the specified pattern. See Chapter 4, "Expressions, Identifiers, and Wildcard Characters" in *Reference Manual: Building Blocks* for details about using wildcard characters.
- Executing sp_helpdb *dbname* from *dbname* includes free space and segment information in the report.
- sp_helpdb displays information about a database's attributes, giving the attribute's class, name, integer value, character value, and comments, if any attributes are defined. Example 3 shows cache binding attributes for the pubs2 database.
- sp_helpdb reports if a database is offline.
- sp_helpdb reports row lock promotion thresholds, if any are defined for the database.
- A database created with the for load option has a status of "don't recover" in the output from sp_helpdb.
- When Component Integration Services is enabled, sp_helpdb lists the default storage location for the specified database or all databases. If there is no default storage location, the display indicates "NULL".

Permissions

Any user can execute sp_helpdb.

See also

Commands alter database, create database

System procedures sp_configure, sp_dboption, sp_rename

sp_helpdevice

Description

Reports information about a particular device or about all Adaptive Server database devices and dump devices.

Syntax

sp_helpdevice [*devname*]

Parameters

devname
is the name of the device about which to report information. If you omit this parameter, sp_helpdevice reports on all devices.

Examples

Example 1 Displays information about all the devices on Adaptive Server:

```
sp_helpdevice

device_name  physical_name   description
----------- -------------  -------------------------------------------
diskdump     null            disk, dump device
master       d_master        special, default disk, dsync on,physical
                                                         disk, 10 MB
status       cntrltype          device_number            low       high
------       ----------         -------------            ----      -----
16           2                  0                        0         20000
3            0                  0                        0         5120
```

Example 2 Reports information about the dump device named diskdump:

```
sp_helpdevice diskdump
```

Usage

- sp_helpdevice displays information on the specified device, when *devname* is given, or on all devices in master.dbo.sysdevices, when no argument is given.
- The sysdevices table contains dump devices and database devices.

  Database devices can be designated as 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, the “cntrltype” is usually 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. sp_helpdevice may report erroneous negative numbers for device numbers greater than 126.

The “low” and “high" columns represent virtual page numbers, each of which is unique among all the devices in Adaptive Server.

Permissions

Any user can execute sp_helpdevice.

See also

Commands disk init, dump database, dump transaction, load database, load transaction

System procedures sp_addumpdevice, sp_deviceattr, sp_diskdefault, sp_dropdevice, sp_logdevice

sp_helpextendedproc

Description

Displays extended stored procedures (ESPs) in the current database, along with their associated DLL files.

Syntax

sp_helpextendedproc [*esp_name*]

Parameters

esp_name
is the name of the extended stored procedure. It must be a procedure in the current database.

Examples

Example 1 Lists the xp_cmdshell ESP and the name of the DLL file in which its function is stored:

```
use sybsystemprocs
go
sp_helpextendedproc xp_cmdshell

ESP Name    DLL Name
----------- ----------
xp_cmdshell sybsyesp
```

Example 2 Lists all the ESPs in the current database, along with the names of the DLL files in which their functions are stored:

```
sp_helpextendedproc

ESP Name    DLL Name
----------- ----------
xp_freedl    sybsyesp
xp_cmdshell  sybsyesp
```

Usage

- If the *esp_name* is omitted, sp_helpextendedproc lists all the extended stored procedures in the database.
- The *esp_name* is case sensitive. It must match the *esp_name* used to create the ESP.

Permissions

Only a System Administrator can execute sp_helpextendedproc to see all the ESPs in the database. All users can execute sp_helpextendedproc to see ESPs owned by themselves or by the Database Owner.

See also

Commands create procedure, drop procedure

Extended system procedure xp_cmdshell

System procedures sp_addextendedproc, sp_dropextendedproc

sp_helpexternlogin

Description

Component Integration Services only Reports information about external login names.

Syntax

sp_helpexternlogin [*remote_server*] [, *login_name*] [*, role_name*]

Parameters

remote_server
is the name of the remote server that has been added to the local server with sp_addserver.

login_name
is a login account on the local server.

role_name
is the Adaptive Server user's assigned role.

Examples

Example 1 Displays all remote servers, local login names, role names, and external logins:

```
sp_helpexternlogin
```

Example 2 Displays local login names, role names, and external logins for the server named SSB:

```
sp_helpexternlogin SSB
```

Example 3 Displays remote servers, local login names and external logins for the user named "milo":

```
sp_helpexternlogin NULL, milo
```

Example 4 Displays external logins for remote server SSB where the local user name is "trixi":

```
sp_helpexternlogin SSB, trixi
```

Example 5 Displays external logins for remote server SSB for local users with sa_role:

```
sp_helpexternlogin SSB, NULL, sa_role
```

Usage

- sp_helpexternlogin displays all remote servers, the user's local login name, role name, and the user's external login name.
- Add remote servers with sp_addserver. Add local logins with sp_addlogin.

Permissions

Any user can execute sp_helpexternlogin.

See also

System procedures sp_addexternlogin, sp_addlogin, sp_addserver, sp_dropexternlogin, sp_helpserver

sp_helpgroup

Description

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

Example 1 Displays information about all groups in the current database:

```
sp_helpgroup

Group_name          Group_id
---------------     --------
hackers             16384
public               0:
```

Example 2 Displays information about the group "hackers":

```
sp_helpgroup hackers

Group_name     Group_id     Users_in_group     Userid
-----------    ---------    --------------     ------
hackers        16384        ann                4
hackers        16384        judy               3
```

Usage

- To get a report on the default group, "public," enclose the name "public" in single or double quotes ("public" is a reserved word).
- If there are no members in the specified group, sp_helpgroup displays the header, but lists no users, as follows:

```
Group_name          Group_id       Users_in_group        Userid
-----------         ---------      --------------        ------
```

Permissions

Any user can execute sp_helpgroup.

See also

Commands grant, revoke

System procedures sp_addgroup, sp_changegroup, sp_dropgroup, sp_helprotect, sp_helpuser

sp_helpindex

Description Reports information about the indexes created on a table.

Syntax sp_helpindex *objname*

Parameters *objname*
is the name of a table in the current database.

Examples **Example 1** Displays the types of indexes on the sysobjects table:

```
sp_helpindex sysobjects

index_name           index_description
      index_keys
      index_max_rows_per_page index_fillfactor index_reservepagegap
 -------------------- ---------------------------------------------------
        --------------------- --------------- --------------------
 sysobjects           clustered, unique located on system
     id
                             0                0                    0
 ncsysobjects         nonclustered, unique located on system
     name,uid
                             0                0                    0
```

Example 2 The index on publ_ix was created with pub_id in ascending order and pubdate in descending order:

```
sp_helpindex titles

index_name       index_description
   index_keys
   index_max_rows_per_page index_fillfactor index_reservepagegap
---------------- ---------------------------------------------
title_id_ix      nonclustered, unique located on default
    title_id
                        0                0                    0
publ_ix          nonclustered located on default
    pub_id, pubdate DESC
                        0                0                    8
title_ix         clustered, allow duplicate rows located on default
    title
                        0               90                    0
```

Usage

- sp_helpindex lists any indexes on a table, including indexes created by defining unique or primary key constraints defined by a create table or alter table statement.

- sp_helpindex displays any attributes (for example, cache bindings) assigned to the indexes on a table.
- sp_helpindex displays:
    - The max_rows_per_page setting of the indexes.
    - Information about clustered indexes on data-only locked tables

      The index ID (indid) of a clustered index in data-only locked tables is not equal to 1.
    - The column order of the keys, to indicate whether they are in ascending or descending order.
    - Space manage property values.
    - The key column name followed by the order. Only descending order is displayed. For example, if there is an index on column a ASC, b DESC, c ASC, “index_keys” shows “a, b DESC, c”.

Permissions

Any user can execute sp_helpindex.

See also

Commands create index, drop index, update statistics

System procedures sp_help, sp_helpkey

sp_helpjava

Description

Displays information about Java classes and associated JARs that are installed in the database.

Syntax

sp_helpjava ["class" [, *java_class_name* [, “detail” | “depends”]] |
    "jar" [, *jar_name* [, “depends”]]]

Parameters

"class" | "jar"
: specifies whether to display information about a class or a JAR. Both “class” and “jar” are keywords, so the quotes are required.

java_class_name
: the name of the class about which you want information. The class must be a system class or a user-defined class that is installed in the database.

detail
: specifies that you want to see detailed information about the class.

depends
: lists all the database objects that depend on the specified class or classes in the JAR, including SQLJ functions, SQLJ stored procedures, views, Transact-SQL stored procedures, and tables.

jar_name
: the name of the JAR for which you want to see information. The JAR must be installed in the database using installjava.

Examples

Example 1 Displays the names of all classes and associated JAR files installed in the database:

```
sp_helpjava
```

Example 2 Displays the name of all classes:

```
sp_helpjava "class"
```

Example 3 Displays detailed information about the Address class:

```
sp_helpjava "class", Address, detail
Class
--------------------------------------------------
Address

(1 row affected)
Class Modifiers
--------------------------------------------------
 public synchronized

 Implemented Interfaces
 --------------------------------------------------
 java.io.Serializable

 Extended Superclass
 --------------------------------------------------
 java.lang.Object

 Constructors
 --------------------------------------------------
 public Address()
 public Address(java.lang.String,java.lang.String)

 Methods
 --------------------------------------------------
 public final native java.lang.Class java.lang.Object.getClass()
 public native int java.lang.Object.hashCode()
 public boolean java.lang.Object.equals(java.lang.Object)
 public java.lang.String java.lang.Object.toString()
 public final native void java.lang.Object.notify()
 public final native void java.lang.Object.notifyAll()
 public final native void java.lang.Object.wait(long) throws
java.lang.InterruptedException
 public final void java.lang.Object.wait(long,int) throws
java.lang.InterruptedException
 public final void java.lang.Object.wait() throws
java.lang.InterruptedException
 public java.lang.String Address.display()
 public void Address.removeLeadingBlanks()

 Fields
 ----------------------------------------
 public java.lang.String Address.street
 public java.lang.String Address.zip
```

Usage

- The depends parameter lists dependencies of a class or classes if the class is listed in the external name clause of a create statement for a SQLJ routine or is used as a datatype of a column in the database.

Permissions

Any user can execute sp_helpjava.

See also

Documents See *Java in Adaptive Server Enterprise* for more information about Java in the database.

Commands remove java

Utilities extractjava, installjava

sp_helpjoins

Description

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

Example 1 Displays a list of columns that are likely join candidates in the sales and salesdetail tables:

```
sp_helpjoins sales, salesdetail
```

```
a1       a2       b1       b2       c1       c2
   d1       d2       e1       e2       f1       f2
      g1       g2       h1       h2
-------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- --------
      -------- -------- -------- --------
stor_id  stor_id  ord_num  ord_num  NULL     NULL
   NULL     NULL     NULL     NULL     NULL     NULL
      NULL     NULL     NULL     NULL
```

Example 2 Displays a list of columns that are likely join candidates in the sysobjects and syscolumns system tables:

```
sp_helpjoins sysobjects, 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
```

Usage

- The column pairs that sp_helpjoins displays come from either of two sources. sp_helpjoins checks the syskeys table in the current database to see if any foreign keys have been defined with sp_foreignkey on the two tables, then checks to see if any common keys have been defined with sp_commonkey on the two tables. If sp_helpjoins does not find any foreign keys or common keys there, it checks for keys with the same user-defined datatypes. If that fails, it checks for columns with the same name and datatype.
- sp_helpjoins does not create any joins.

Permissions	Any user can execute sp_helpjoins.
See also	**System procedures** sp_commonkey, sp_foreignkey, sp_helpkey, sp_primarykey

sp_helpkey

Description

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 [*tabname*]

Parameters

tabname

is the name of a table or view in the current database. If you do not specify a name, the procedure reports on all keys defined in the current database.

Examples

Displays information about the keys defined in the current database. The "object_keys" and "related_keys" columns refer to the names of the columns that make up the key:

```
sp_helpkey

keytype object      related_object object_keys         related_keys
------- -------     -------------- ---------------     --------------
primary authors     -- none --     au_id,*,*,*,*,*,*,*  *,*,*,*,*,*,*,*
foreign titleauthor authors        au_id,*,*,*,*,*,*,* au_id,*,*,*,*,*,
                                                               *,*
```

Usage

- sp_helpkey lists information about all primary, foreign, and common key definitions that reference the table *tabname* or, if *tabname* is omitted, about all the keys in the database. 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. 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 so that applications can use the information.
- If you specify an object name, sp_helpkey follows the Adaptive Server rules for finding objects:
  - If you do not specify an owner name, and you own an object with the specified name, sp_helpkey reports on that object.
  - If you do not specify an owner name, and you do not own an object of that name, but the Database Owner does, sp_helpkey reports on the Database Owner's object.
  - If neither you nor the Database Owner owns an object with the specified name, sp_helpkey reports an error condition, even if an object with that name exists in the database for a different owner.

    - If both you and the Database Owner own objects with the specified name, and you want to access the Database Owner's object, specify the name in the form *dbo.objectname*.

- Qualify objects that are owned by database users other than yourself and the Database Owner with the owner's name, as in "mary.myproc".

Permissions

Any user can execute sp_helpkey.

See also

Commands create trigger

System procedures sp_commonkey, sp_foreignkey, sp_primarykey

sp_helplanguage

Description — Reports information about a particular alternate language or about all languages.

Syntax — sp_helplanguage [*language*]

Parameters — *language*
is the name of the alternate language you want information about.

Examples — **Example 1** Displays information about the alternate language, “french”:

```
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
```

Example 2 Displays information about all installed alternate languages:

```
sp_helplanguage
```

Usage

- sp_helplanguage reports on a specified language, when the language is given, or on all languages in master.dbo.syslanguages, when no language is supplied.

Permissions — Any user can execute sp_helplanguage.

See also — **System procedures** sp_addlanguage, sp_droplanguage, sp_setlangalias

sp_helplog

Description

Reports the name of the device that contains the first page of the transaction log.

Syntax

sp_helplog

Parameters

None.

Examples

```
sp_helplog
```

```
In database 'master', the log starts on device 'master'.
```

Usage

- sp_helplog displays the name of the device that contains the first page of the transaction log in the current database.

Permissions

Any user can execute sp_helplog.

See also

Commands alter database, create database

System procedures sp_helpdevice, sp_logdevice

sp_helpobjectdef

Description

Component Integration Services only Reports owners, objects, and type information for remote object definitions.

Syntax

sp_helpobjectdef [*object_name*]

Parameters

object_name
is the name of the object as it is defined in the sysattributes table. The *object_name* can be in any of the following forms:

- *dbname.owner.object*
- *dbname..object*
- *owner.object*
- *object*

dbname and *owner* are optional. *object* is required. If *owner* is not supplied, the *owner* defaults to the current user name. If *dbname* is supplied, it must be the current database, and *owner* must be supplied or marked with the placeholder *dbname..object*. Enclose a multipart *object_name* in quotes.

Examples

Example 1 Displays all remote object definitions in the current database:

```
sp_helpobjectdef
```

Example 2 Displays remote object definitions for the tb1 table owned by the Database Owner:

```
sp_helpobjectdef "dbo.tb1"
```

Usage

- If no *object_name* is supplied, sp_helpobjectdef displays all remote object definitions.
- A server name is not permitted in the *object_name* parameter.

Permissions

Any user can execute sp_helpobjectdef.

See also

Commands create table, create existing table, drop table

System procedures sp_addobjectdef, sp_dropobjectdef, sp_helpserver

sp_help_qpgroup

Description

Reports information on an abstract plan group.

Syntax

sp_help_qpgroup [*group* [, *mode*]]

Parameters

group

is the name of an abstract plan group.

mode

is the type of report to print, one of the following:

Mode	Information returned
full	The number of rows and number of plans in the group, the number of plans that use two or more rows, the number of rows and plan IDs for the longest plans, and number of hash keys and hash key collision information. This is the default report mode.
stats	All of the information from the “full” report, except hash key information.
hash	The number of rows and number of abstract plans in the group, the number of hash keys, and hash-key collision information.
list	The number of rows and number of abstract plans in the group, and the following information for each query/plan pair: hash key, plan ID, first few characters of the query, and the first few characters of the plan.
queries	The number of rows and number of abstract plans in the group, and the following information for each query: hash key, plan ID, first few characters of the query.
plans	The number of rows and number of abstract plans in the group, and the following information for each plan: hash key, plan ID, first few characters of the plan.
counts	The number of rows and number of abstract plans in the group, and the following information for each plan: number of rows, number of characters, hash key, plan ID, first few characters of the query.

Examples

Example 1 Reports summary information about all abstract plan groups in the database:

```
sp_help_qpgroup

Group                       GID         Plans
----------------------- ----------- -----------
ap_stdin                          1           0
ap_stdout                         2           0
dev_test                          3         209
```

Example 2 Reports on the test_plans group:

```
sp_help_qpgroup test_plans

Query plans group ’test_plans’, GID 8

 Total Rows  Total QueryPlans
 ----------- ---------------
           6               3
sysqueryplans rows consumption, number of query plans
per row count

 Rows        Plans
 ----------- -----------
           2           3

 Hashkeys
 -----------
           3
There is no hash key collision in this group.
```

Usage

- When used with an abstract plan group name, and no mode parameter, the default mode for sp_help_qpgroup is full.
- Hash-key collisions indicate that more than one plan for a particular user has the same hash-key value. When there are hash key collisions, the query text of each query with the matching hash key must be compared to the user's query text in order to identify the matching query, so performance is slightly degraded.

Permissions

Any user can execute sp_help_qpgroup.

See also

System procedures sp_help_qplan

sp_help_qplan

Description
Reports information about an abstract plan.

Syntax
sp_help_qplan *id* [, *mode*]

Parameters
id
is the ID of the abstract plan.

mode
is the type of report to print, one of the following:

mode	Information returned
full	The plan ID, group ID, and hash key, and the full query and plan text.
brief	The same as full, but only prints about 80 characters of the query and plan, rather than the full query and plan. This is the default mode.
list	The hash key, ID, and first 20 characters of the query and plan.

Examples
Example 1 Prints the brief abstract plan report:

```
sp_help_qplan 800005881
gid         hashkey     id
 ----------- ----------- -----------
           5  2054169974   937054374

 query
---------------------------------------------------------------------------
 select type, avg(price) from titles group by type

 plan
---------------------------------------------------------------------------
 ( plan
    ( store Worktab1
        ( i_scan type_price titles )
    )
    ( t_scan ( ...
```

Example 2 Prints the full abstract plan report:

```
sp_help_qplan 784005824, full
```

Usage
- If you do not supply a value for the mode parameter, the default is brief.

Permissions
Any user can execute sp_help_qplan to see the abstract plan of a query that he or she owns. Only the System Administrator and the Database Owner can display an abstract plan owned by another user.

See also
System procedures sp_find_qplan, sp_help_qpgroup

sp_helpremotelogin

Description	Reports information about a particular remote server's logins or about all remote server 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

Example 1 Displays information about all the remote users of the remote server GATEWAY:

```
sp_helpremotelogin GATEWAY
```

Example 2 Displays information about all the remote users of all the remote servers known to the local server:

```
sp_helpremotelogin
```

Usage

- sp_helpremotelogin reports on the remote logins for the specified server, when *remoteserver* is given, or on all servers, when no parameter is supplied.

Permissions

Any user can execute sp_helpremotelogin.

See also

System procedures sp_addremotelogin, sp_dropremotelogin, sp_helpserver

sp_help_resource_limit

Description

Reports on resource limits.

Syntax

sp_help_resource_limit [*name* [, *appname* [, *limittime*
    [, *limitday* [, *scope* [, *action*]]]]]]

Parameters

name

is the Adaptive Server login to which the limits apply. For information about limits that govern a particular login, specify the login *name*. For information about limits without regard to login, specify null.

Note If you are not a System Administrator, specify your own login, or a login of NULL, to display information about the resource limits that apply to you.

appname

is the name of the application to which the limit applies. For information about limits that govern a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet. For information about limits without regard to application, specify null.

limittime

is the time during which the limit is enforced. For information about limits in effect at a given time, specify the time, with a value between "00:00" and "23:59", using the following form:

```
"HH:MM"
```

For information about limits without regard to time, specify null.

limitday

is any day on which the limit is enforced. For information about resource limits in effect on a given day of the week, specify the full weekday name for the default server language, as stored in the syslanguages system table of the master database. For information about limits without regard to the days on which they are enforced, specify null.

scope

is the scope of the limit. Specify one of the following:

Scope code	For help on all limits that govern
1	Queries
2	Query batches (one or more SQL statements sent by the client to the server)
4	Transactions
6	Both query batches and transactions

Scope code	For help on all limits that govern
NULL	The specified *name*, *appname*, *limittime*, *limitday*, and *action*, without regard to their *scope*

action
is the action to take when the limit is exceeded. Specify one of the following:

Action code	For help on all limits that
1	Issue a warning
2	Abort the query batch
3	Abort the transaction
4	Kill the session
NULL	Govern the specified *name*, *appname*, *limittime*, *limitday*, and *scope*, without regard to the *action* they take

Examples

Example 1 Lists all resource limits stored in the sysresourcelimits system table:

```
sp_help_resource_limit
```

Example 2 Lists all limits for the user "joe_user":

```
sp_help_resource_limit joe_user
```

Example 3 Lists all limits for the application *my_app*:

```
sp_help_resource_limit NULL, my_app
```

Example 4 Lists all limits enforced at 9:00 a.m.:

```
sp_help_resource_limit NULL, NULL, "09:00"
```

Example 5 An alternative way of listing the limits enforced at 9:00 a.m.:

```
sp_help_resource_limit @limittype = "09:00"
```

Example 6 Lists all limits enforced on Mondays:

```
sp_help_resource_limit NULL, NULL, NULL, Monday
```

Example 7 Lists any limit in effect for "joe_user" on Mondays at 9:00 a.m.:

```
sp_help_resource_limit joe_user, NULL, "09:00", Monday
```

Usage

- sp_help_resource_limit reports on all resource limits, limits for a given login or application, limits in effect at a given time or day of the week, or limits with a given scope or action.

Permissions

Any user can execute sp_help_resource_limit to list his or her own resource limits. Only a System Administrator can execute sp_help_resource_limit to list limits that apply to other users.

See also

Documents See the *System Administration Guide* for more information on resource limits.

System procedures sp_add_resource_limit, sp_drop_resource_limit, sp_modify_resource_limit

sp_helprotect

Description

Reports on permissions for database objects, users, groups, or roles.

Syntax

```
sp_helprotect [name [, username [, "grant"
        [,"none"|"granted"|"enabled"|role_name]]]]
```

Parameters

name
is either the name of the table, view, stored procedure, SQLJ stored procedure, SQLJ function, or the name of a user, user-defined role, or group in the current database. If you do not provide a name, sp_helprotect reports on all permissions in the database.

username
is a user's name in the current database.

grant
displays the privileges granted to *name* with grant option.

none
ignores roles granted to the user when determining permissions granted.

granted
includes information on all roles granted to the user when determining permissions granted.

enabled
includes information on all roles activated by the user when determining permissions granted.

role_name
displays permission information for the specified role only, regardless of whether this role has been granted to the user.

Examples

Example 1 This series of grant and revoke statements, executing sp_helprotect titles results in this display:

```
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
```

```
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
```

Example 2 Issuing the following grant statement results in sp_helprotect displaying the following:

```
grant select, update on titles(price, advance)
   to mary
   with grant option
go
sp_helprotect titles
```

```
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
```

Example 3 Displays all the permissions that "judy" has in the database:

```
sp_helprotect judy
```

Example 4 Displays any permissions that "csmith" has on the sysusers table, as well as whether "csmith" has with grant option which allows "csmith" to grant permissions to other users:

```
sp_helprotect sysusers, csmith, null, doctor, "grant"
```

```
grantor   grantee   type   action     object        column   grantable
-----------------------------------------------------------------
dbo       doctor    Grant  Delete     sysusers      All     FALSE
dbo       doctor    Grant  Insert     sysusers      All     FALSE
dbo       doctor    Grant  References sysusers      All     FALSE
dbo       doctor    Grant  Select     sysattributes All     FALSE

(1 row affected)
(return status = 0)
```

Example 5 Displays information about the permissions that the doctor role has in the database:

```
sp_helprotect doctor_role
```

```
grantor   grantee   type   action     object          column   grantable
-----------------------------------------------------------------
dbo       doctor     Grant  Delete     sysusers        All    FALSE
dbo       doctor     Grant  Insert     sysusers        All    FALSE
dbo       doctor     Grant  References sysusers        All    FALSE
dbo       doctor     Grant  Select     sysattributes   All    FALSE

(1 row affected)
(return status = 0)
```

Example 6 Displays information on all roles granted to "csmith":

```
sp_helprotect sysusers, csmith, null, doctor_role, "granted"
```

```
grantor grantee   type   action     object    column   grantable
-----------------------------------------------------------------
dbo     csmith    Grant   Update     sysusers   All     FALSE
dbo     doctor    Grant   Delete     sysusers   All     FALSE
dbo     doctor    Grant   Insert     sysusers   All     FALSE
dbo     doctor    Grant   References sysusers   All     FALSE

(1 row affected)
(return status = 0)
```

Example 7 Displays information on all active roles granted to "rpillai":

```
sp_helprotect sysattributes, rpillai, null, intern, "enabled"
```

```
grantor   grantee   type   action   object        column  grantable
--------------- -----------------------------------------------
dbo       public     Grant  Select   sysattributes All     FALSE

(1 row affected)
(return status = 0)
```

Example 8 Advises that SQLJ function access is public:

```
sp_helprotect function_sqlj
```

```
Implicit grant to public for SQLJ functions.
```

Usage

- sp_helprotect reports permissions on a database object. If you supply the *username* parameter, only that 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, a group, or a role. If it is, sp_helprotect lists the permissions for the user, group, or role.
- sp_helprotect looks for objects and users in the current database only.

- If you do not specify an optional value such as granted, enabled, none, or *role_name*, Adaptive Server returns information on all roles activated by the current specified user.
- If the specified user is not the current user, Adaptive Server returns information on all roles granted to the specified user.
- Displayed information always includes permissions granted to the group in which the specified user is a member.
- In granting permissions, a System Administrator is treated as the object owner. If a System Administrator grants permission on another user's object, the owner's name appears as the grantor in sp_helprotect output.

Permissions

Any user can execute sp_helprotect to view his or her own permissions. Only a System Security Officer can execute sp_helprotect to view permissions granted to other users.

See also

Commands grant, revoke

System procedures sp_activeroles, sp_displayroles

sp_helpsegment

Description

Reports information about a particular segment or about all 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 about all segments in the current database appears.

Examples

Example 1 Reports information about all segments in the current database:

```
sp_helpsegment

segment name                           status
------- ------------------------------ ------
      0 system                              0
      1 default                             1
      2 logsegment                          0
```

Example 2 Reports information about the segment named order_seg, including which database tables and indexes use that segment and the total number of pages, free pages and used pages on the segment:

```
sp_helpsegment order_seg

segment name                           status
------- ------------------------------ ------
      3 order_seg                        0

device                 size          free_pages
---------------------- ------------- -----------
tpcd_data1             25.0MB               8176
tpcd_data2             25.0MB               8512
tpcd_data3             25.0MB               8392
tpcd_data4             25.0MB               8272
tpcd_data5             25.0MB               8448
tpcd_data6             25.0MB               8512

table_name             index_name             indid
---------------------- ---------------------- ------
orders                 orders                  0

total_size     total_pages free_pages  used_pages
-------------- ----------- ----------- -----------
150.0MB              76800       50312       26488
```

Example 3 Reports information about the default segment. The keyword default must be enclosed in quotes:

```
sp_helpsegment "default"
```

Example 4 Reports information about the segment on which the transaction log is stored:

```
sp_helpsegment logsegment

segment name                          status
------- ------------------------------ ------
      2 logsegment                         0

device                 size              free_pages
---------------------- ---------------- -----------
tpcd_log1              20.0MB                 10200

table_name             index_name                indid
---------------------- ------------------------- ------
syslogs                syslogs                     0

total_size        total_pages free_pages  used_pages
----------------- ----------- ----------- -----------
20.0MB                  10240       10200          40
```

Usage

- sp_helpsegment displays information about the specified segment, when *segname* is given, or about all segments in the current database, when no argument is given.
- When you first create a database, Adaptive Server automatically creates the system, default, and logsegment segments. Use sp_addsegment to add segments to the current database.
- If you specify a log segment from a dedicated log database for the *segname* parameter, sp_helpsegment reports the number of free pages in the log segment.
- The system, default, and logsegment segments are numbered 0, 1, and 2, respectively.
- The "status" column indicates which segment is the default pool of space. Use sp_placeobject or the on *segment_name* clause of the create table or create index command to place objects on specific segments.
- The "indid" column is 0 if the table does not have a clustered index and is 1 if the table has a clustered index.

Permissions

Any user can execute sp_helpsegment.

See also

Commands create index, create table

System procedures sp_addsegment, sp_dropsegment, sp_extendsegment, sp_helpdb, sp_helpdevice, sp_placeobject

sp_helpserver

Description

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 about which you want information.

Examples

Example 1 Displays information about the remote server GATEWAY:

```
sp_helpserver GATEWAY
```

Example 2 Displays information about the local Backup Server:

```
sp_helpserver SYB_BACKUP

name       network_name  status                              id
---------- -------------- ----------------------------------- ---
SYB_BACKUP SYB_BACKUP    timeouts, no net password encryption  1
```

Example 3 Displays information about all the remote servers known to the local server:

```
sp_helpserver
```

Usage

- sp_helpserver reports information about all servers in master.dbo.sysservers or about a particular remote server, when *server* is specified.
- When Component Integration Services is installed, sp_helpserver lists the server class for each server.

Permissions

Any user can execute sp_helpserver.

See also

System procedures sp_addserver, sp_dropserver, sp_helpremotelogin, sp_serveroption

sp_helpsort

Description Displays Adaptive Server's default sort order and character set.

Syntax sp_helpsort

Parameters None.

Examples 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:

```
sp_helpsort

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 °  1/4 1/2 3/4 ¿ 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.
```

Usage

- Binary sort order is the default.

Permissions

Any user can execute sp_helpsort.

sp_helptext

Description

Displays the **source text** of a **compiled object**.

Syntax

sp_helptext *objname [,number]*

Parameters

objname

is the name of the compiled object for which the source text is to be displayed. The compiled object must be in the current database.

number

is an integer identifying an individual procedure, when *objname* represents a group of procedures. This parameter tells sp_helptext to display the source text for a specified procedure in the group.

Note Views, defaults, and other non-procedural objects are never grouped; use *number* only for groups of procedures.

Examples

Example 1 Displays the source text of pub_idrule. Since this rule is in the pubs2 database, execute this command from pubs2:

```
sp_helptext pub_idrule
```

```
# Lines of Text
---------------
1

text
------------------------------------
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877",
    "1622", "1756")
    or @pub_id like "99[0-9][0-9]"
```

Example 2 Displays the source text of sp_helptext. Since system procedures are stored in sybsystemprocs, execute this command from sybsystemprocs:

```
sp_helptext sp_helptext
```

Example 3 Displays the source text of the *myproc* group behavior where you specify no *number* argument. The number of the procedure displays beside the text:

```
sp_helptext myproc
```

```
# Lines of Text
---------------
2
```

```
number
text
---------------
1
create procedure myproc; as select 1
2
create procedure myproc;2 as select 2
(2 rows affected)
```

Example 4 Displays the source text of *myproc*, specifying a procedure in the *myproc* group but displaying no grouping number.

```
sp_helptext myproc, 2
```

```
# Lines of Text
---------------
1
text
----------------
create procedure myproc;2 as select 2
```

Usage

- sp_helptext prints out the number of rows in syscomments (255 characters long each) that are occupied by the compiled object, followed by the source text of the compiled object.
- sp_helptext looks for the source text in the syscomments table in the current database.
- You can encrypt the source text with sp_hidetext.
- When sp_helptext operates on a group of procedures, it prints the number column from syscomments in addition to the source text.
- A System Security Officer can prevent the source text of compiled objects from being displayed to most users who execute sp_helptext. To restrict select permission on the text column of the syscomments table to the object owner or a System Administrator, use sp_configure to set the select on syscomments.text column parameter to 0. This restriction is required to run Adaptive Server in the evaluated configuration. See the *System Administration Guide* for more information about the evaluated configuration.

Permissions

Any user can execute sp_helptext.

See also

System procedures sp_checksource, sp_configure, sp_hidetext

sp_helpthreshold

Description	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 [*segname*]
Parameters	*segname* is the name of a segment in the current database.

Examples

Example 1 Shows all thresholds on the log segment:

```
sp_helpthreshold logsegment
```

Example 2 Shows all thresholds on all segments in the current database:

```
sp_helpthreshold
```

Example 3 Shows all thresholds on the default segment. Note the use of quotes around the reserved word "default":

```
sp_helpthreshold "default"
```

Usage

- sp_helpthreshold displays threshold information for all segments in the current database. If you provide the name of a segment, sp_helpthreshold lists all thresholds in 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.

Permissions

Any user can execute sp_helpthreshold.

See also

System procedures sp_addthreshold, sp_dropthreshold, sp_helpsegment, sp_modifythreshold, sp_thresholdaction

sp_helpuser

Description

Reports information about a particular user, group, or alias, 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

Example 1 Displays information about all users in the current database:

```
sp_helpuser

Users_name ID_in_db    Group_name  Login_name
---------  --------    ----------  ----------
ann        4           hackers     ann
dbo        1           public      sa
guest      2           public      NULL
judy       3           hackers     judy
```

Example 2 Displays information about the Database Owner (user name "dbo"):

```
sp_helpuser dbo

Users_name     ID_in_db    Group_name  Login_name
----------     --------    ----------  ----------
dbo            1           public      sa
Users aliased to user.
Login_name
-------------------------------
andy
christa
howard
linda
```

Usage

- sp_helpuser reports information about all users of the current database. If you specify a *name_in_db*, sp_helpuser reports information on the specified user only.
- 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.

Permissions

Any user can execute sp_helpuser.

See also

System procedures sp_adduser, sp_dropuser, sp_helpgroup

sp_hidetext

Description	Hides the source text for the specified compiled object.
Syntax	sp_hidetext [*objname* [, *tabname* [, *username*]]]

Parameters

objname
specifies the compiled object for which to hide the source text.

tabname
specifies the name of the table or view for which to hide the source text.

username
specifies the name of the user who owns the compiled object for which to hide the source text.

Examples

Example 1 Hides the source text of all compiled objects in the current database:

```
sp_hidetext
```

Example 2 Hides the source text of the user-defined stored procedure, sp_sort_table, that is owned by Mary:

```
sp_hidetext @objname = "sp_sort_table", @username = "Mary"
```

Example 3 Hides the source text of the stored procedure pr_phone_list:

```
sp_hidetext "pr_phone_list"
```

Example 4 Hides the source text of all check constraints, defaults, and triggers defined on the table my_tab:

```
sp_hidetext @tabname = "my_tab"
```

Example 5 Hides the source text of the view my_vu and all check constraints, defaults, and triggers defined on the table my_tab:

```
sp_hidetext "my_vu", "my_tab"
```

Example 6 Hides the source text of all compiled objects that are owned by Tom:

```
sp_hidetext @username = "Tom"
```

Usage

- sp_hidetext hides the source text for the specified compiled object.

  Warning! Before executing sp_hidetext, make sure you have a backup of the source text. The results of executing sp_hidetext are not reversible.

- If you do not provide any parameters, sp_hidetext hides the source text for all compiled objects in the current database.

Permissions	Any user can use sp_hidetext to hide the source text of his or her own compiled objects. Only a Database Owner or a System Administrator can hide the source text of compiled objects that are owned by another user or use sp_hidetext with no parameters.
See also	**Documents** See the *Transact-SQL User's Guide* for more information about hiding source text.
	System procedures sp_checksource

sp_import_qpgroup

Description

Imports abstract plans from a user table into an abstract plan group.

Syntax

sp_import_qpgroup *tab*, *usr*, *group*

Parameters

tab

is the name of a table from which to copy the plans. You can specify a database name, but not an owner name, in the form *dbname..tablename*. The total length must be 30 characters or less.

usr

is the name of the user whose ID should be assigned to the abstract plans when they are imported.

group

is the name of the abstract plan group that contains the plans to be imported.

Examples

Copies plans from the table moveplans to the new_plans group, giving them the user ID for the Database Owner:

```
sp_import_qpgroup moveplans, dbo, new_plans
```

Usage

- sp_import_qpgroup copies plans from a user table to an abstract plan group in sysqueryplans. With sp_export_qpgroup, it can be used to copy abstract plan groups between servers and databases, or to copy plans belonging to one user and assign them the ID of another user.
- sp_import_qpgroup creates the abstract plan group if it does not exist when the procedure is executed.
- If an abstract plan group exists when sp_import_qpgroup is executed, it cannot contain any plans for the specified user. sp_import_qpgroup does not check the query text to determine whether queries already exist in the group. If you need to import plans for a user into a group where some plans for the user already exist:
    - Use sp_import_qpgroup to import the plans into a new plan group.
    - Use sp_copy_all_qplans to copy the plans from the newly-created group to the destination group. sp_copy_all_qplans does check queries to be sure that no duplicate plans are created.
    - If you no longer need the group you created for the import, drop the plans in the group with sp_copy_all_qplans, then drop the group with sp_drop_qpgroup.
- To create an empty table in order to bulk copy abstract plans, use:

```
select * into load_table
```

```
from sysqueryplans
where 1 = 2
```

Permissions

Only a System Administrator or the Database Owner can execute sp_import_qpgroup.

See also

Commands create plan

System procedures sp_copy_all_qplans, sp_copy_qplan, sp_drop_all_qplans, sp_drop_qpgroup, sp_export_qpgroup, sp_help_qpgroup

sp_indsuspect

Description

Checks user tables for indexes marked as suspect during recovery following a sort order change.

Syntax

sp_indsuspect [*tab_name*]

Parameters

tab_name
is the name of the user table to be checked.

Examples

Checks the table newaccts for indexes marked as suspect:

```
sp_indsuspect newaccts
```

Usage

- sp_indsuspect with no parameter creates a list of all tables in the current database that have indexes that need to be rebuilt as a result of a sort order change. With a *tab_name* parameter, sp_indsuspect checks the specified 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.

Permissions

Any user can execute sp_indsuspect.

See also

Commands dbcc

sp_ldapadmin

Description

Creates an LDAP URL search string, lists an LDAP URL search string, or verifies an LDAP URL search string or login.

Syntax

```
sp_ldapadmin { set_primary_url, 'ldapurl' |
          set_secondary_url, { 'ldapurl' | null } |
          list_urls | check_url, 'ldapurl' |
          check_login, 'login_name' }
```

ldapurl::=ldap://*host*:*port*/*node*/?*attributes*?base | one | sub?*filter*

Parameters

set_primary_url, '*ldapurl*'
creates the specified search string *ldapurl*. Exactly one primary search string can be created.

set_secondary_url, { '*ldapurl*' | null }
creates the specified secondary search string *ldapurl* or no secondary search string. Exactly one secondary search string can be created.

list_urls
displays LDAP URL search strings.

check_url, '*ldapurl*'
verifies an LDAP URL search string. Can also verify the existence of a user account, but it does not authenticate the user.

check_login, *login_name*
verifies a user account for the existing LDAP URL search strings. It does not authenticate the user.

host
is the host name of the LDAP server.

port
is the port number of the LDAP server.

node
specifies the node in the object hierarchy at which to start the search.

attributes
is a list of attributes to return in the result set. Each LDAP server may support a different list of attributes.

base | one | sub
qualifies the search criteria. base specifies a search of the base node; one specifies a search of node and one sublevel below node; and sub specifies a search of node and all node sublevels.

filter

specifies the attribute or attributes to be authenticated. The filter can be simple, such as “uid=*,” or compound, such as “(uid=*)(ou=*group*).” The syntax is LDAP server dependent and uses a wildcard (*) to describe the login name.

Examples

Example 1 Creates an LDAP URL search string for the LDAP SunONE Directory Server.

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
    ou=People,dc=MyCompany,dc=com??sub?uid=*'
```

The search string identifies a directory server listening on host name “voyager,” port number 389 (the default LDAP protocol port), the base node to begin the search is within organizational unit (ou) “People,” and the domain is “MyCompany.com.” It returns all attributes that match the filter uid=*. Adaptive Server replaces the wildcard with the Adaptive Server login name that is to be authenticated.

Example 2 Creates an LDAP URL search string defined in OpenLDAP 2.0.25 using the criteria described in Example 1.

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
    dc=MyCompany,dc=com??sub?cn=*'
```

Example 3 Sets the secondary LDAP URL search string to null, indicating no failover and no secondary LDAP server.

```
sp_ldapadmin set_secondary_url, null
```

Example 4 Creates an LDAP URL search string with a compound filrer.

```
sp_ldapadmin set_primary_url, 'ldap://voyager:389/
    ou=people,dc=siroe,dc=com??sub?(&(uid=*)
    (ou=accounting))
```

Usage

- The LDAP vendor determines the syntax of the search string. In all cases, the search string specifies the attribute name that uniquely identifies the user in the form “*attribute*=*wildcard*” as in “cn=*.”
- The first attribute in a compound filter must define the Relative Distinguished Name (RDN). For example, “...sub?(uid=*)(ou=group).” Otherwise, the authentication fails.
- When a search string is added, Adaptive Server verifies that it uses valid LDAP URL syntax and that it references an existing node. To ensure that the valid string returns expected values, carefully choose and verify the search string when configuring Adaptive Server.

- The secondary URL search string enables failover to another LDAP server. Adaptive Server uses the primary URL search string unless the LDAP Server is not active or the search string is invalid. In this event, Adaptive Server uses the secondary URL search string for authentication.

Permissions

Only the System Security Officer can execute sp_ldapadmin.

sp_listener

Description

Dynamically starts and stops listeners on Adaptive Server on any given port on a per-engine basis.

Syntax

sp_listener "*command*", "*server_name*", *engine* | *remaining*

Or:

sp_listener "*command*", "*[protocol:]machine:port*", *engine*

Parameters

command

can be any of the following:

- start – starts a listener on the specified ports on each of the specified engines
- stop – terminates the specified listeners.
- suspend – prevent the listener from accepting any more connections.
- resume – instructs suspended listeners to resume listening again.
- status – report on the state of the listeners specified by the parameters. The state is one of: active, stopped, or suspended.

server_name

is the name of the Adaptive Server, as specified in the interfaces file.

engine

specifies the number of the engine affected by this command (this parameter is ignored by Windows NT. *engine* can be a single-engine number in quotes ("2"), a list ("3,5,6"), a range ("2-5"), or mix of all ("2,3-5,7")

remaining

specifies that the command is to take effect on all engines on which it can be meaningfully applied (that is, where the listener is in a state in which the command is can take effect).

protocol

the type of protocol. This can be one of: tcp, tli, ssltcp, ssltli, winsock, sslnlwnsck, sslwinsock.

machine:port

the machine name and port number (as specified in the interfaces file) to which the listener connects.

Examples

Example 1 Create tcp listeners on engines 0-6 for port number 4226:

```
sp_listener "start", "goldie:4226", "0-6"
```

Example 2 Create listeners for all master entries in the interfaces file for server orion:

```
sp_listener "start", "orion", "remaining"
```

Example 3 Start listeners on engines 1, 3 and 5 for each master entry in the interfaces file corresponding to server orion:

```
sp_listener "start", "orion", "1,3,5"
```

Example 4 Start tcp listeners on port 4226 on machine goldie for all engines not already listening to this port:

```
sp_listener "start", "goldie:4226", "remaining"
```

Example 5 Stop the listener on port number 4226 on engine number 2:

```
sp_listener "stop", "tcp:goldie:4226", "2"
```

Example 6 Stop all listeners on port number 4226 for all engines. Because this command includes the remaining parameter, it will not fail if some engines are not listening to the port:

```
sp_listener "stop", "tcp:goldie:4226", "remaining"
```

Example 7 Suspend NT winsock listener on port 4226:

```
sp_listener "suspend", "winsock:clouds:4226"
```

Example 8 Resume NT winsock listener on port 4227:

```
sp_listener "suspend", "winsock:clouds:4226"
```

Example 9 Resume all active listeners on port number 4226:

```
sp_listener "resume", "tcp:goldie:4226", "remaining"
```

Usage

- sp_listener uses either of two syntaxes, described in the syntax section, above. The first syntax affects all Adaptive Server master ports listed in the interfaces file. The second allows you to manage listeners not listed in the interfaces file.
- The semantics for sp_listener is atomic: if a command cannot be completed successfully, it is aborted.
- You can issue the status parameter by itself. The status parameter displays the state of all the listeners in the interfaces file.
- A listener can be in one of the following states: stopped, suspended, or active. sp_listener allows you to move listeners between these states. A request to move to a non-permissible state results in failure (For example, requesting to stop a non existent listener). Use sp_listener “status” to determine the state of a listener.

- You can specify engines in the engine list as: an engine number (a single interger in quotes), a range of engine numbers (“3-6”), a comma separated list of engines (“2,5,9”), or combinations of the previous two (“2,5,3-6,9”).
- The remaining parameter specifies that, for the command you are running (start, stop, resume, and so on), the command runs successfully for all listeners that are in a state that allow the change (for example, moving states from start to stop). For example, if you attempt to start listeners on engines one through six, but engines one, four, and five are unavailable, sp_listener... “remaining” starts listeners on engines two, three, and six, disregarding the offline engines. You cannot specify an engine number if you include the remaining parameter.
- The maximum number of listeners is 32. If you create an Adaptive Server with two master ports in the interfaces file, you can start at most 30 more listeners on other ports. Apart from the first listener, each supplementary listener consumes resources similar to a user connection, so in a setup where 25 user connections are configured, starting three listeners at startup (corresponding to three master entries) leaves room for 30 user connections.

sp_listsuspect_db

Description	Lists all databases that currently have offline pages because of corruption detected on recovery.
Syntax	sp_listsuspect_db
Parameters	None.

Examples

Lists the databases that have suspect pages:

```
sp_listsuspect_db
```

Usage

- sp_listsuspect_db lists the database name, number of suspect pages, and number of objects containing suspect pages.
- Use sp_listsuspect_page to identify the suspect pages.

Permissions

Any user can execute sp_listsuspect_db.

See also

System procedures sp_listsuspect_page, sp_setsuspect_granularity, sp_setsuspect_threshold

sp_listsuspect_object

Description

Lists all indexes in a database that are currently offline because of corruption detected on recovery.

Syntax

sp_listsuspect_object [*dbname*]

Parameters

dbname
is the name of the database.

Examples

Example 1 Lists the suspect indexes in the current database:

```
sp_listsuspect_object
```

Example 2 Lists the suspect indexes in the pubs2 database:

```
sp_listsuspect_object pubs2
```

Usage

- If an index on a data-only-locked table has suspect pages, the entire index is taken offline during recovery. Offline indexes are not considered by the query optimizer.
- Use the system procedure sp_forceonline_object to bring an offline index online for repair.
- Indexes on allpages-locked tables are not taken completely offline during recovery; only individual pages of these indexes are taken offline. These pages can be brought online with sp_forceonline_object.
- sp_listsuspect_object lists the database name, object ID, object name, index ID, and access status for every suspect index in the specified database or, if *dbname* is omitted, in the current user database.
- A value of SA_ONLY in the access column means that the index has been forced online for System Administrator use only. A value of BLOCK_ALL means that the index is offline for everyone.

Permissions

Any user can execute sp_listsuspect_object.

See also

Documents See the *System Administration Guide* for more information on recovery fault isolation.

System procedures sp_forceonline_object

sp_listsuspect_page

Description

Lists all pages in a database that are currently offline because of corruption detected on recovery.

Syntax

sp_listsuspect_page [*dbname*]

Parameters

dbname
is the name of the database.

Examples

Example 1 Lists the suspect pages in the current database:

```
sp_listsuspect_page
```

Example 2 Lists the suspect pages in the pubs2 database:

```
sp_listsuspect_page pubs2
```

Usage

- sp_listsuspect_page lists the database name, page ID, object, index ID, and access status for every suspect page in the specified database or, if *dbname* is omitted, in the current user database.
- A value of SA_ONLY in the "access" column indicates that the page has been forced online for System Administrator use only. A value of BLOCK_ALL indicates that the page is offline for everyone.

Permissions

Any user can execute sp_listsuspect_page.

See also

System procedures sp_listsuspect_db, sp_setsuspect_granularity, sp_setsuspect_threshold

sp_lock

Description

Reports information about processes that currently hold locks.

Syntax

sp_lock [*spid1* [, *spid2*]]

Parameters

spid1

is the Adaptive Server process ID number from the master.dbo.sysprocesses table. Run sp_who to get the *spid* of the locking process.

spid2

is another Adaptive Server process ID number to check for locks.

Examples

Example 1 This example shows the lock status of serial processes with spids 7, 18, and 23 and two families of processes. The family with fid 1 has the coordinating processes with spid 1 and worker processes with spids 8, 9, and 10. The family with fid 11 has the coordinating processes with spid 11 and worker processes with spids 12, 13, and 14:

```
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.


fid spid locktype        table_id  page  dbname  class           context
--- ---- ------------- ---------- ----- ------- --------------  -------
0    7  Sh_intent      480004741     0 master  Non Cursor Lock NULL
0   18  Ex_intent       16003088     0 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088   587 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088   590 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088  1114 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088  1140 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088  1283 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088  1362 pubtune Non Cursor Lock NULL
0   18  Ex_page         16003088  1398 pubtune Non Cursor Lock NULL
0   18  Ex_page-blk     16003088   634 pubtune Non Cursor Lock NULL
0   18  Update_page     16003088  1114 pubtune Non Cursor Lock NULL
0   18  Update_page-blk 16003088   634 pubtune Non Cursor Lock NULL
0   23  Sh_intent       16003088     0 pubtune Non Cursor Lock NULL
0   23  Sh_intent      176003658     0 pubtune Non Cursor Lock NULL
0   23  Ex_intent      208003772     0 pubtune Non Cursor Lock NULL
1    1   Sh_intent     176003658    0  tpcd   Non Cursor Lock Sync-pt
duration request
1    1   Sh_intent-blk 208003772    0  tpcd   Non Cursor Lock Sync-pt
duration request
1    8  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
1    9  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
1   10  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
```

```
11  11   Sh_intent     176003658     0 tpcd   Non Cursor Lock Sync-pt
duration request
11  12  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
11  13  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
11  14  Sh_page        176003658 41571 tpcd    Non Cursor Lock NULL
```

Example 2 Displays information about the locks currently held by spid 7.

```
sp_lock 7
```

```
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.
fid spid locktype   table_id  page dbname  class            context
--- ---- ---------  --------- ---- ------  ---------------- ----------
 0   7   Sh_intent  480004741    0 master  Non Cursor Lock    NULL
```

Usage

- 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 system function to derive a table's name from its ID number.
- sp_lock output is ordered by fid and then spid.
- The loid column identifies unique lock owner ID of the blocking transaction. Even loid values indicate that a local transaction owns the lock. Odd values indicate that an external transaction owns the lock.
- The locktype column indicates whether the lock is a shared lock ("Sh" prefix), an exclusive lock ("Ex" prefix) or an update lock, and whether the lock is held on a table ("table" or "intent") or on a page ("page").

  A "blk" suffix in the "locktype" column indicates that this process is blocking another process that needs to acquire a lock. As soon as this process completes, the other process(es) moves forward. A "demand" suffix in the "locktype" column indicates that the process is attempting to acquire an exclusive lock. See the *Performance and Tuning Guide* for more information about lock types.
- The class column indicates whether a lock is associated with a cursor. It displays one of the following:
  - "Non Cursor Lock" indicates that the lock is not associated with a cursor.

- “Cursor Id *number*” indicates that the lock is associated with the cursor ID number for that Adaptive Server process ID.
- A cursor name indicates that the lock is associated with the cursor *cursor_name* that is owned by the current user executing sp_lock.

- The fid column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for fid are:
  - A zero value indicates that the task represented by the spid is executed serially. It is not participating in parallel execution.
  - A nonzero value indicates that the task (spid) holding the lock is a member of a family of processes (identified by fid) executing a statement in parallel. If the value is equal to the spid, it indicates that the task is the coordinating process in a family executing a query in parallel.
- The context column identifies the context of the lock. Worker processes in the same family have the same context value. Legal values for “context” are as follows:
  - “NULL” means that the task holding this lock is either a query executing serially, or is a query executing in parallel in transaction isolation level 1.
  - “Sync-pt duration request” means that the task holding the lock will hold the lock until the query is complete.

    A lock’s context may be “Sync-pt duration request” if the lock is a table lock held as part of a parallel query, if the lock is held by a worker process at transaction isolation level 3, or if the lock is held by a worker process in a parallel query and must be held for the duration of the transaction.
  - “Ind pg” indicates locks on index pages (allpages-locked tables only)
  - “Inf key” indicates an infinity key lock (for certain range queries at transaction isolation level 3 on data-only-locked tables)
  - “Range” indicates a range lock (for range queries at transaction isolation level 3 on data-only-locked tables)

  These new values may appear in combination with “Fam dur” (which replaces “Sync pt duration”) and with each other, as applicable.
- The row column displays the row number for row-level locks.
- sp_lock output also displays the following lock types:

- "Sh_row" indicates shared row locks
- "Update_row" indicates update row locks
- "Ex_row" indicates exclusive row locks

Permissions	Any user can execute sp_lock.
See also	**Commands** kill, select
	System procedures sp_familylock, sp_who

sp_locklogin

Description

Locks an Adaptive Server account so that the user cannot log in or displays a list of all locked accounts.

Syntax

sp_locklogin [*loginame*, "{lock | unlock}"]

Parameters

loginame
is the name of the account to be locked or unlocked.

lock | unlock
specifies whether to lock or unlock the account.

Examples

Example 1 Locks the login account for the user “charles”:

```
sp_locklogin charles, "lock"
```

Example 2 Displays a list of all locked accounts:

```
sp_locklogin
```

Usage

- Locking an Adaptive Server login account prevents that user from logging in. Use sp_locklogin instead of sp_droplogin for the following reasons:
  - You cannot drop a login who is a user in any database, and you cannot drop a user from a database if the user owns any objects in that database or has granted any permissions on objects to other users.
  - Adaptive Server may reuse the dropped login account’s server user ID (suid) when the next login account is created. This occurs only when the dropped login holds the highest suid in syslogins; however, it could compromise accountability if execution of sp_droplogin is not being audited. In addition, it is possible that the user with the reused suid will actually be able to access database objects that were authorized for the old suid.
  - You cannot drop the last remaining System Security Officer’s or System Administrator’s login account.
- sp_locklogin with no parameters returns a list of all the locked accounts.
- 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.

Permissions

Only a System Security Officer can execute sp_locklogin.

See also

System procedures sp_addlogin, sp_droplogin, sp_modifylogin, sp_password

sp_logdevice

Description

Moves the transaction log of a database with log and data on the same device to a separate database device.

Syntax

sp_logdevice *dbname*, *devname*

Parameters

dbname
is the name of the database whose syslogs table, which contains the transaction log, to put on a specific logical device.

devname
is the logical name of the device on which 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

Example 1 Creates the database products and puts the table products.syslogs on the database device logs:

```
create database products on default = "10M", logs = "2M"
go
sp_logdevice products, logs
go
```

Example 2 For the database test with log and data on the same device, places the log for test on the log device logdev:

```
alter database test log on logdev
go
sp_logdevice test, logdev
go
```

Usage

- 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 the use of 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, noncritical database could keep its log together with the rest of the database. Such databases use dump database to back up 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, depending on the amount of update activity and the frequency of transaction log dumps. As a rule, allocate to the log device 10 percent to 25 percent of the space you allocate to the database itself.
- Use sp_logdevice only for a database with log and data on the same device. Do not use sp_logdevice for a database with log and data on separate devices.
- 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 the following to increase the size of the log segment. If you did not use log on, execute sp_logdevice:

  ```
  sp_extendsegment segname, devname
  ```

  The device or segment on which you put syslogs is used *only* for the syslogs table. 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.

Permissions

Only the Database Owner or a System Administrator can execute sp_logdevice.

See also

Documents See the *System Administration Guide* for more information.

Commands alter database, create database, dbcc, disk init, dump database, dump transaction, select

System procedures sp_extendsegment, sp_helpdevice, sp_helplog

sp_loginconfig

Description

Windows NT only Displays the value of one or all integrated security parameters.

Syntax

sp_loginconfig ["*parameter_name*"]

Parameters

parameter_name
is the name of the integrated security parameter you want to examine. Values are:

- login mode
- default account
- default domain
- set host
- key _
- key $
- key @
- key #

Examples

Example 1 Displays the values of all integrated security parameters:

```
sp_loginconfig

name                   config_item
---------------------- ----------------------
login mode             standard
default account        NULL
default domain         NULL
set host               false
key _                  domain separator
key $                  space
key @                  space
key #                  -
```

Example 2 Displays the value of the login mode security parameter:

```
sp_loginconfig "login mode"

name                   config_item
---------------------- ----------------------
login mode             standard
```

Usage

- The values of integrated security parameters are stored in the Windows NT Registry. See the chapter on login security in *Configuration Guide for Windows NT* for instructions on changing the parameters.
- sp_loginconfig displays the *config_item* values that were in effect when you started Adaptive Server. If you changed the Registry values after starting Adaptive Server, those values are not reflected in the sp_loginconfig output.

Permissions

Only a System Administrator can execute sp_loginconfig.

See also

System procedures sp_revokelogin

sp_logininfo

Description

Windows NT only Displays all roles granted to Windows NT users and groups with sp_grantlogin.

Syntax

sp_logininfo ["*login_name*" | "*group_name*"]

Parameters

login_name
is the network login name of the Windows NT user.

group_name
is the Windows NT group name.

Examples

Example 1 Displays the permissions granted to the Windows NT user "regularjoe":

```
sp_logininfo regularjoe

account name    mapped login name  type            privilege
--------------- ------------------ --------------- --------------
HAZE\regularjoe HAZE_regularjoe    user            'oper_role'
```

Example 2 Displays all permissions that were granted to Windows NT users and groups with sp_grantlogin:

```
sp_logininfo

account name            mapped login name       type        privilege
---------------         -------------------     ------------  ------------
BUILTIN\Administrators  BUILTIN\Administrators  group
        'sa_role sso_role oper_role sybase_ts_role navigator_role
              replication_role'
HAZE\regularjoe         HAZE_regularjoe        user         'oper_role'
PCSRE\randy             PCSRE_alexander        user         'default'
```

Usage

- sp_logininfo displays all roles granted to Windows NT users and groups with sp_grantlogin.
- You can omit the domain name and domain separator (\) when specifying the Windows NT user name or group name.

Permissions

Only a System Administrator can execute sp_logininfo.

See also

Commands grant, setuser

System procedures sp_displaylogin, sp_grantlogin, sp_revokelogin, sp_role, sp_who

sp_logiosize

Description

Changes the log I/O size used by Adaptive Server to a different memory pool when doing I/O for the transaction log of the current database.

Syntax

sp_logiosize ["default" | "*size*" | "all"]

Parameters

default
: sets the log I/O size for the current database to Adaptive Server's default value (two logical pages), if a memory pool that is two logical pages is available in the cache. Otherwise, Adaptive Server sets the log I/O size to one logical page. Since default is a keyword, the quotes are required when specifying this parameter.

size
: is the size to set the log I/O for the current database. Values are multiples of the logical page size, up to four times the amount. You must enclose the value in quotes.

all
: displays the log I/O size configured for all databases grouped by the cache name.

Examples

Example 1 Displays the log I/O size configured for the current database:

```
sp_logiosize

The transaction log for database 'master' will use I/O size of 2 Kbytes.
```

Example 2 Changes the log I/O size of the current database to use the 8K memory pool. If the database's transaction log is bound to a cache that does not have an 8K memory pool, Adaptive Server returns an error message indicating that such a pool does not exist, and the current log I/O size does not change:

```
sp_logiosize "8"
```

Example 3 Changes the log I/O size of the current database to Adaptive Server's default value (one logical page size). If a memory pool the size of the logical page size does not exist in the cache used by the transaction log, Adaptive Server uses the 2K memory pool:

```
sp_logiosize "default"
```

Example 4 Displays the log I/O size configured for all databases:

```
sp_logiosize "all"

Cache name: default data cache
Data base                       Log I/O Size
------------------------------  ------------
master                          2 Kb
```

```
tempdb                          2 Kb
model                           2 Kb
sybsystemprocs                  2 Kb
pubs3                           2 Kb
pubtune                         2 Kb
dbccdb                          2 Kb
sybsyntax                       2 Kb
```

Usage

- sp_logiosize displays or changes the log I/O size for the current database. Any user can execute sp_logiosize to display the configured log I/O size. Only a System Administrator can change the log I/O size.
- If you specify sp_logiosize with no parameters, Adaptive Server displays the log I/O size of the current database.
- When you change the log I/O size, it takes effect immediately. Adaptive Server records the new I/O size for the database in the sysattributes table.
- Any value you specify for sp_logiosize must correspond to an existing memory pool configured for the cache used by the database's transaction log. Specify these pools using the sp_poolconfig system procedure.

  Adaptive Server defines the default log I/O size of a database as two logical pages, if a memory pool the size of two logical pages is available in the cache. Otherwise, Adaptive Server sets the log I/O size to one logical page (a memory pool of one logical page is always present in any cache). For most work loads, a log I/O size of two logical pages performs much better than one of one logical page, so each cache used by a transaction log should have a memory pool the size of a logical page. See the *System Administration Guide* and the *Performance and Tuning Guide* for more information about configuring caches and memory pools.
- If the transaction logs for one or more databases are bound to a cache of type logonly, any memory pools in that cache that have I/O sizes larger than the log I/O size defined for those databases will *not* be used.

  For example, on a 2K server, assume that only two databases have their transaction logs bound to a "log only" cache containing 2K, 4K, and 8K memory pools. By default, sp_logiosize sets the log I/O size for these parameters at 4K, and the 8K pool is not used. Therefore, to avoid wasting cache space, be cautious when configuring the log I/O size.

- During recovery, only the logical page size memory pool of the default cache is active, regardless of the log I/O size configured for a database. Transactions logs are read into this pool of the default cache, and all transactions that must be rolled back, or rolled forward, read data pages into the default data cache.

Permissions

Only a System Administrator can execute sp_logiosize to change the log I/O size for the current database. Any user can execute sp_logiosize to display the log I/O size values.

See also

System procedures sp_cacheconfig, sp_poolconfig

sp_modifylogin

Description

Modifies the default database, default language, default role activation, login script, full name, the password expiration interval, the minimum password length, and the maximum number of failed logins allowed for a specified Adaptive Server login account.

Syntax

sp_modifylogin {*loginame* | “all overrides”}, *option*, *value*

Parameters

loginame
is the login account to be modified.

"all overrides"
removes the system overrides that were set using the "passwd expiration", "min passwd length", or "max failed_logins" parameters. To remove all the login-specific values, specify:

```
sp_modifylogin "all overrides", "option", "-1"
```

option
specifies the name of the option to be changed. 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.
"add default role"	The role or roles to be activated by default at login.
"drop default role"	The role or roles to be dropped from the list of roles activated by default at login. This option affects only user-defined roles, not system roles.
"passwd expiration"	The password expiration interval in days. It can be anyvalue between 0 and 32767, inclusive.
"min passwd length"	The minimum password length required for the specified login. It can be any value between 0 and 30, inclusive. 0 specifies that no password is required. The default is 6.
"max failed_logins"	The number of allowable failed login attempts for the specified login. It can be any value between 0 and 32767, inclusive.
login script	Name of the stored prodecure to run automatically when user logs in.

value
is the value of the option you specified for the *option* parameter. The *value* parameter is a character datatype; therefore, quotes are required for positive and negative numeric values.

Examples

Example 1 Changes the default database for “sarah” to pubs2:

```
sp_modifylogin sarah, defdb, "pubs2"
```

Example 2 Sets the default language for “claire” to French:

```
sp_modifylogin claire, deflanguage, "french"
```

Example 3 Changes the full name of user “clemens” to “Samuel Clemens”:

```
sp_modifylogin clemens, fullname, "Samuel Clemens"
```

Example 4 Adds the specialist role to the list of roles activated by default when user csmith logs in:

```
sp_modifylogin csmith, "add default role", specialist_role
```

Example 5 Drops the intern role from the list of roles activated by default when user “hpillai” logs in:

```
sp_modifylogin hpillai, "drop default role", intern_role
```

Example 6 Changes the maximum number of failed login attempts for the login “joe” to 40:

```
sp_modifylogin "joe", "max failed_logins", "40"
```

Example 7 Changes the overrides for maximum failed login attempts of all logins to 3:

```
sp_modifylogin "all overrides", "max failed_logins", "3"
```

Example 8 Removes the overrides for maximum failed logins option for all logins:

```
sp_modifylogin "all overrides", "max failed_logins", "-1"
```

Example 9 Runs the *proc_p1* script when Bob1 logs in to Adaptive Server:

```
sp_modifylogin Bob1, 'login script', proc_p1
```

Usage

- Set a default database, language, or full name either with sp_modifylogin or with sp_addlogin when first adding the user’s login to Adaptive 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.
- If there are any login triggers associated with the login in question, they are listed after the `Auto Login Script` line. For more information, see in “Row-level access control” in Chapter 11, “Managing User Permissions” of the *System Administration Guide*.

- You cannot use double quotes in the script name for the login script option. For example, Adaptive Server issues an error message it you specify a login script named *"script"name"*.
- For more information about password expiration interval, minimum password length, and maximum number of failed logins, see "User-Defined Login Security" in the *System Administration Guide*.

Changing a user's default database

- After sp_modifylogin is executed to change the user's default database, the user is connected to the new *defdb* the next time he or she logs in. However, the user cannot access the database until the Database Owner gives the user access through sp_adduser or sp_addalias, or unless 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.

Changing a user's role activation

- Use sp_modifylogin to set a role to be activated by default at login or to drop a role from those activated by default at login.

Permissions

A user can use sp_modifylogin to change their default database, default language, and full name. Only a System Administrator or System Security Officer can execute sp_modifylogin to change the default database, default language, or full name of another user. Only a System Security Officer can execute sp_modifylogin to activate another user's roles by default at login, change the password expiration interval, the minimum password length, the maximum number of failed logins allowed, and the login script associated with a specified login.

See also

System procedures sp_activeroles, sp_addalias, sp_addlogin, sp_adduser, sp_displaylogin, sp_displayroles, sp_helprotect,

Commands alter role, create procedure, create role, drop role, grant, revoke, set role

sp_modify_resource_limit

Description

Changes a resource limit by specifying a new limit value, or the action to take when the limit is exceeded, or both.

Syntax

```
sp_modify_resource_limit {name, appname }
    rangename , limittype , limitvalue , enforced , action , scope
```

Parameters

name
is the Adaptive Server login to which the limit applies. You must specify either a *name* or an *appname* or both. To modify a limit that applies to all users of a particular application, specify a *name* of null.

appname
is the name of the application to which the limit applies. You must specify either a *name* or an *appname* or both. If the limit applies to all applications used by *name*, specify an *appname* of null. If the limit governs a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename
is the time range during which the limit is enforced. You cannot modify this value, but you must specify a non-null value to uniquely identify the resource limit.

limittype
is the type of resource to which the limit applies. You cannot modify this value, but you must specify a non-null value to uniquely identify the resource limit. The value must be one of the following:

Limit type	Description
row_count	Limits the number of rows a query can return
elapsed_time	Limits the number of seconds in wall-clock time that a query batch or transaction can run
io_cost	Limits either the actual cost, or the optimizer's cost estimate, for processing a query
tempdb_space	Limits the number of pages a tempdb database can have during a single session

limit_value
is the maximum amount of the server resource that the login or application can use before Adaptive Server enforces the limit. This must be a positive integer less than or equal to 2^{31} or null to retain the existing value. The following table indicates what value to specify for each limit type:

Limit type	Limit value
row_count	The maximum number of rows a query can return before the limit is enforced
elapsed_time	The maximum number of seconds in wall-clock time that a query batch or transaction can run before the limit is enforced

Limit type	Limit value
io_cost	A unitless measure derived from optimizer's costing formula
tempdb_space	The number of pages used in tempdb per session.

enforced

determines whether the limit is enforced prior to or during query execution. You cannot modify this value. Use null as a placeholder.

action

is the action to take when the limit is exceeded. The following codes apply to all limit types:

Action code	Description
1	Issues a warning
2	Aborts the query batch
3	Aborts the transaction
4	Kills the session
null	Retains the existing value

scope

is the scope of the limit. You cannot modify this value. You can use null as a placeholder.

Examples

Example 1 Modifies a resource limit that applies to all applications used by "robin" during the *weekends* time range. The limit issues a warning when a query is expected to return more than 3000 rows:

```
sp_modify_resource_limit robin, NULL, weekends, row_count, 3000, NULL,
    1, NULL
```

Example 2 Modifies a resource limit that applies to the *acctg* application on all days of the week and at all times of the day. The limit aborts the query batch when estimated query processing time exceeds 45 seconds:

```
sp_modify_resource_limit NULL, acctg, "at all times", elapsed_time,
    45, 2, 2, 6
```

Usage

- You cannot change the login or application to which a limit applies or specify a new time range, limit type, enforcement time, or scope.
- The modification of a resource limit causes the limits for each session for that login and/or application to be rebound at the beginning of the next query batch for that session.

Permissions

Only a System Administrator can execute sp_modify_resource_limit.

See also

Documents For more information, see the *System Administration Guide*.

System procedures sp_add_resource_limit, sp_drop_resource_limit, sp_help_resource_limit

sp_modify_time_range

Description

Changes the start day, start time, end day, and/or end time associated with a named time range.

Syntax

sp_modify_time_range *name, startday*, *endday*, *starttime*, *endtime*

Parameters

name

is the name of the time range. This must be the name of a time range stored in the systimeranges system table of the master database.

startday

is the day of the week on which the time range begins. This must be the full weekday name for the default server language, as stored in the syslanguages system table of the master database, or null to keep the existing *startday*.

endday

is the day of the week on which the time range ends. This must be the full weekday name for the default server language, as stored in the syslanguages system table of the master database, or null to keep the existing end day. The *endday* can fall either earlier or later in the week than the *startday*, or it can be the same day as the *startday*.

starttime

is time of day at which the time range begins. Specify the *starttime* in terms of a twenty-four hour clock, with a value between 00:00 and 23:59. Use the following form, or null to keep the existing *starttime*:

```
"HH:MM"
```

endtime

is the time of day at which the time range ends. Specify the *endtime* in terms of a twenty-four hour clock, with a value between 00:00 (midnight) and 23:59. Use the following form, or null to keep the existing *endtime*:

```
"HH:MM"
```

The *endtime* must occur later in the day than the *starttime,* unless *endtime* is 00:00.

Note For time ranges that span the entire day, specify a start time of “00:00” and an end time of “23:59”.

Examples

Example 1 Changes the end day of the *business_hours* time range from Friday to Saturday. Retains the existing start day, start time, and end time:

```
sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL
```

Example 2 Specifies a new end day and end time for the *before_hours* time range:

```
sp_modify_time_range before_hours, Monday, Saturday, NULL, "08:00"
```

Usage

- You cannot modify the “at all times” time range.
- It is possible to modify a time range so that it overlaps with one or more other time ranges.
- The modification of time ranges through the system stored procedures does not affect the active time ranges for sessions currently in progress.
- Changes to a resource limit that has a transaction as its scope does not affect any transactions currently in progress.

Permissions

Only a System Administrator can execute sp_modify_time_range.

See also

Documents For more information, see the *System Administration Guide*.

System procedures sp_add_resource_limit, sp_add_time_range, sp_drop_time_range

sp_modifystats

Description

Allows the System Administrator to modify the density values of a column—or columns—in sysstatistics

Syntax

```
sp_modifystats [database].[owner].table_name,
          {"column_group" | "all"},
          MODIFY_DENSITY,
          {range | total},
          {absolute | factor},
          "value"
```

Or,

```
    sp_modifystats [database].[owner].table_name,
          column_name,
          REMOVE_SKEW_FROM_DENSITY
```

Parameters

table_name

is the name of the table to change. Specify the database name if the table is in another database, and specify the owner's name if more than one table of that name exists in the database. The default value for *owner* is the current user, and the default value for *database* is the current database.

column_group

an ordered list of column names. To change a statistic for multiple columns (such as a density value), list the columns in the order used to create the statistic. Separate the column names with commas. For example, if your table has a density statistic on columns a1, a2, a3, a4:

- "`a1`" modifies column a1.
- "`a1,a2,a3`" modifies the column group a1,a2,a3,
- You can also use a wildcard character , %, with the column_group parameter to represent a range of characters. For example, "`a1,%,a3`" modifies the groups a1,a2,a3 and a1, a4, a3, and so on; "`a1,%`" modifies the groups a1,a2 and a1,a2,a3, and so on, but not a1; "`a1%`" modifies the groups a1,a2 and a1,a2,a3, and so on, as well as a1.

all

modifies all column group for this table. Because "all" is a keyword, it requires quotes.

MODIFY_DENSITY
allows you to modify either the range or total density of a column or column group to the granularity specified in the *value* parameter. Range cell density represents the average number of duplicates of all values that are represented by range cells in a histogram. See the *Performance and Tuning Guide* for more information.

range
modifies the range cell density.

total
modifies the total cell density.

absolute
ignore the current value and use the number specified by the *value* parameter.

factor
multiply the current statistical value by the *value* parameter.

value
is either the specified density value or a multiple for the current density. Must be between zero and one, inclusive, if absolute is specified.

column_name
is the name of a column in that table.

REMOVE_SKEW_FROM_DENSITY
allows the System Administrator to change the total density of a column to be equal to the range density, which is useful when data skew is present. Total density represents the average number of duplicates for all values, those in both frequency and range cells. Total density is used to estimate the number of matching rows for joins and for search arguments whose value is not known when the query is optimized. See the *Performance and Tuning Guide* for more information.

REMOVE_SKEW_FROM_DENSITY also updates the total density of any composite column statistics for which this column is the leading attribute. Most commonly, a composite index for which this column is the leading attribute would produce these composite column statistics, but they can also be produced when you issue a composite update statistics command.

Examples

Example 1 Changes the range density for column group c00, c01 in table tab_1 to 0.50000000:

```
sp_modifystats  "tab_1", "c00, c01", MODIFY_DENSITY, range, absolute, "0.5"
```

Example 2 The total density for column group c00, c01 in tab_1 is multiplied by .5. That is, divided in half:

```
sp_modifystats  "tab_1", "c00,c01", MODIFY_DENSITY, total, factor, "0.5"
```

Example 3 The total density for all the columns in table tab_1 is multiplied by .5.

```
sp_modifystats  "tab_1", "all", MODIFY_DENSITY, total, factor, "0.5"
```

Example 4 Total density for all column groups starting with c12 is changed to equal the range density.

```
sp_modifystats "tab_1", "c12" REMOVE_SKEW_FROM_DENSITY
```

Usage

- Allows the System Administrator to modify the density values of a column—or columns—in sysstatistics.
- Use optdiag to view a table's statistics. See the *Performance and Tuning Guide* for more information about table density and using optdiag.
- Any modification you make to the statistics with sp_modifystats is overwritten when you run update statistics. To make sure you are using the most recent statistical modifications, you should run sp_modifystats after you run update statistics.
- Because sp_modifystats modifies information stored in the sysstatistics table, you should make a backup of statistics before execute running sp_modifystats in a production system.

Permissions

No one has default use of sp_modifystats. A person with sso_role must specify the permissions on sp_modifystats.

Tables used

sysstatistics

See also

Command update statistics

sp_modifythreshold

Description

Modifies a threshold by associating it with a different threshold procedure, free-space level, or segment name. You *cannot* use sp_modifythreshold to change the amount of free space or the segment name for the last-chance threshold.

Syntax

sp_modifythreshold *dbname*, *segname*, *free_space*
[, *new_proc_name*] [, *new_free_space*] [, *new_segname*]

Parameters

dbname
is the database for which to change the threshold. This must be the name of the current database.

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

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

new_proc_name
is the new stored procedure to execute when the threshold is crossed. The procedure can be located in any database on the current Adaptive Server or on an Open Server. Thresholds cannot execute procedures on remote Adaptive Servers.

new_free_space
is the new number of free pages to associate with the threshold. When free space in the segment falls below this level, Adaptive Server executes the associated stored procedure.

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

Examples

Example 1 Modifies a threshold on the "default" segment of the mydb database to execute when free space on the segment falls below 175 pages instead of 200 pages. NULL is a placeholder indicating that the procedure name is not being changed:

```
sp_modifythreshold mydb, "default", 200, NULL, 175
```

Example 2 Modifies a threshold on the data_seg segment of mydb so that it executes the new_proc procedure:

```
sp_modifythreshold mydb, data_seg, 250, new_proc
```

Usage

Crossing a threshold

- When a threshold is crossed, Adaptive Server executes the associated stored procedure. Adaptive Server uses the following search path for the threshold procedure:

  - If the procedure name does not specify a database, Adaptive 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_", Adaptive Server looks in the sybsystemprocs database.

  If the procedure is not found in either database, Adaptive Server sends an error message to the error log.

- Adaptive 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, Adaptive 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 Adaptive Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, Adaptive 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 Adaptive 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.
- 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.

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.
- sp_modifythreshold checks to ensure that the user modifying the threshold procedure has been directly granted the "sa_role". All system roles active when the threshold procedure is modified are entered in systhresholds as valid roles for the user writing the procedure. However, only directly granted system roles are activated when the threshold fires. Indirectly granted system roles and user-defined roles are not activated.
- Adaptive Server passes four parameters to a threshold procedure:
    - *@dbname*, varchar(30), which identifies the database
    - *@segment_name*, 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 the procedure 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 can be executed by all thresholds.
- Include print and raiserror statements in the threshold procedure to send output to the error log.

Executing threshold procedures

- Tasks that are 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".
- Adaptive 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

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

- 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.

Permissions

Only the Database Owner or a System Administrator can execute sp_modifythreshold.

See also

Documents For more information, see the *System Administration Guide.*

Commands create procedure, dump transaction

System procedures sp_addthreshold, sp_dboption, sp_dropthreshold, sp_helpthreshold, sp_thresholdaction

sp_monitor

Description Displays statistics about Adaptive Server.

Syntax sp_monitor

Parameters None.

Examples Reports information about how busy Adaptive Server has been:

```
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)
```

Usage

- Adaptive 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 columns) since Adaptive Server restarted.
  - The number in parentheses refers to the number of seconds or the total number since the last time sp_monitor was run. The percent sign indicates the percentage of time since sp_monitor was last run.

For example, if the report shows cpu_busy as “4250(215)-68%”, it means that the CPU has been busy for 4250 seconds since Adaptive Server was last started, 215 seconds since sp_monitor last ran, and 68 percent of the total time since sp_monitor was last run.

For the total_read column, the value 394(67) means there have been 394 disk reads since Adaptive Server was last started, 67 of them since the last time sp_monitor was run.

- Table 1-16 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.

Table 1-16: Columns in the sp_monitor report

Column heading	**Equivalent variable**	**Meaning**
last_run		Clock time at which the sp_monitor procedure last ran.
current_run		Current clock time.
seconds		Number of seconds since sp_monitor last ran.
cpu_busy	*@@cpu_busy*	Number of seconds in CPU time that Adaptive Server’s CPU was doing Adaptive Server work.
io_busy	*@@io_busy*	Number of seconds in CPU time that Adaptive Server has spent doing input and output operations.
idle	*@@idle*	Number of seconds in CPU time that Adaptive Server has been idle.
packets_received	*@@pack_received*	Number of input packets read by Adaptive Server.
packets_sent	*@@pack_sent*	Number of output packets written by Adaptive Server.
packet_errors	*@@packet_errors*	Number of errors detected by Adaptive Server while reading and writing packets.
total_read	*@@total_read*	Number of disk reads by Adaptive Server.
total_write	*@@total_write*	Number of disk writes by Adaptive Server.
total_errors	*@@total_errors*	Number of errors detected by Adaptive Server while reading and writing.
connections	*@@connections*	Number of logins or attempted logins to Adaptive Server.

- The first time sp_monitor runs after Adaptive Server start-up, the number in parentheses is meaningless.

- Adaptive Server's housekeeper task uses the server's idle cycles to write changed pages from cache to disk. This process affects the values of the cpu_busy, io_busy, and idle columns reported by sp_monitor. To disable the housekeeper task and eliminate these effects, set the housekeeper free write percent configuration parameter to 0:

```
sp_configure "housekeeper free write percent", 0
```

Permissions Only a System Administrator can execute sp_monitor.

See also **System procedures** sp_who

sp_monitorconfig

Description

Displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases. sp_monitorconfig also reports statistics on auxiliary scan descriptors used for referential integrity queries, and usage statistics for transaction descriptors and DTX participants.

Syntax

sp_monitorconfig "*configname*" [, “*result_tbl_name*”]

Parameters

configname

is either all, or part of the configuration parameter name whose monitoring information is being queried. Valid configuration parameters are listed in the “Usage” section. Specifying all displays descriptor help information for all indexes, objects, databases, and auxiliary scan descriptors in the server.

“*result_tbl_name*”

is the name of the table you create, in which to save the stored procedure results. This is an optional parameter.

Examples

Example 1

```
sp_monitorconfig "open"

Configuration option is not unique.
option_name                    config_value run_value
------------------------------ ------------ -----------
curread change w/ open cursors            1           1
number of open databases                 12          12
number of open indexes                  500         500
number of open objects                  500         500
open index hash spinlock ratio          100         100
open index spinlock ratio               100         100
open object spinlock ratio              100         100


sp_monitorconfig "all"
-------------

Usage information at date and time: Oct 25 2002 10:36AM.
Name                          num_free  num_active  pct_act  Max_Used  Reu
sed

 ---------------------- ----------- ----------- ------- ----------- ------
additional network memory          0           0   0.00           0     NA
 audit queue size                100           0   0.00           0     NA
 heap memory per user           4096           0   0.00           0     No
 max cis remote connection         0           0   0.00           0     NA
 max memory                    12404       21388  63.29       21388     NA
 max number network listen         3           2  40.00           2     NA
 max online engines                   4           1  20.00          1      NA
 memory per worker process         1024           0   0.00          0      NA
 number of alarms                    31           9  22.50          9      NA
 number of aux scan descri          200           0   0.00          0      NA
 number of devices                    9           1  10.00          1      NA
 number of dtx participant          500           0   0.00          0      NA
 number of java sockets               0           0   0.00          0      NA
 number of large i/o buffers          6           0   0.00          0      NA
 number of locks                   4673         327   6.54        408      NA
 number of mailboxes                 30           0   0.00          0      NA
 number of messages                  64           0   0.00          0      NA
 number of open databases             6           6  50.00          6      No
 number of open indexes             492           8   1.60          8      No
 number of open objects             482          18   3.60         18      No
 number of remote connections        20           0   0.00          0      NA
 number of remote logins             20           0   0.00          0      NA
 number of remote sites              10           0   0.00          0      NA
 number of sort buffers             500           0   0.00          9      NA
 number of user connection           23           2   8.00          2      NA
 number of user processes             0           0   0.00          0      NA
 partition groups                  1024          0   0.00          0      NA
 permission cache entries            15           0   0.00          0      NA
 procedure cache size              2567         704  21.52        810      No
 size of global fixed heap          150           0   0.00          0      NA
 size of process object heap       1500           0   0.00          0      NA
 size of shared class heap         1536           0   0.00          0      NA
 size of unilib cache                 0           0   0.00          0      NA
 txn to pss ratio                    16           0   0.00          0      NA
(return status = 0)
```

Example 2 Shows 283 active object metadata descriptors, with 217 free. The maximum used at a peak period since Adaptive Server was last started is 300:

```
sp_monitorconfig "open objects"

Usage information at date and time: Apr 22 2002  2:49PM.
Name            num_free   num_active  pct_act   Max_Used   Reused
--------------  --------   ----------  -------   --------   ------
number of open  217        283         56.60     300        No
```

You can then reset the size to 330, for example, to accommodate the 300 maximum used metadata descriptors, plus space for 10 percent more:

```
sp_configure "number of open objects", 330
```

Example 3 Shows the maximum number of index metadata descriptors, which is 44:

```
sp_monitorconfig "open indexes"

Usage information at date and time: Apr 22 2002  2:49PM.
Name            num_free   num_active  pct_act   Max_Used   Reused
--------------  --------   ----------  --------  --------   ------
number of open  556         44          7.33      44          No
```

You can reset the size to 100, the minimum acceptable value:

```
sp_configure "number of open indexes", 100
```

Example 4 Shows the number of active scan descriptors as 30, though Adaptive Server is configured to use 200. Use the number of aux scan descriptors configuration parameter to reset the value to at least 32. A safe setting is 36, to accommodate the 32 scan descriptors, plus space for 10 percent more:

```
sp_monitorconfig "aux scan descriptors"

Usage information at date and time: Apr 22 2002  2:49PM.
Name            num_free   num_active  pct_act   Max_Used   Reused
--------------  --------   ----------  --------  --------   ------
number of aux s  170        30          15.00      32         No
```

Example 5 Adaptive Server is configured for five open databases, all of which have been used in the current session.

```
sp_monitorconfig "number of open databases"

Usage information at date and time: Apr 22 2002  2:49PM.
Name            num_free   num_active  pct_act   Max_Used   Reused
--------------  --------   ----------  --------  --------   ------
number of open  0           5           100.00    5           Yes
```

However, as indicated by the Reused column, an additional database needs to be opened. If all 5 databases are in use, an error may result, unless the descriptor for a database that is not in use can be reused. To prevent an error, reset number of open databases to a higher value.

Example 6 Only 10.2 percent of the transaction descriptors are currently being used. However, the maximum number of transaction descriptors used at a peak period since Adaptive Server was last started is 523:

```
sp_monitorconfig "txn to pss ratio"

Usage information at date and time: Apr 22 2002  2:49PM.
Name             num_free    num_active   pct_act    Max_Used    Reused
--------------   --------    ----------   --------   --------    ------
number of open   784         80           10.20      523         NA
```

Example 7 Using the optional parameter *result_tbl_name* to create a user table, saves the sp_monitorconfig result to this table:

```
create table sample_table(
    Name varchar(35), Num_free int,
    Num_active int, Pct_act char(6),
    Max_Used int, Reuse_cnt int,
    Date varchar(30))
```

The name of the table created becomes the second parameter of sp_monitorconfig:

```
sp_monitorconfig "number of alarms", sample_table

-------------
(return status = 0)

select * from sample_table

---------------
Name              Num_free  Num_active  Pct_act  Max_Used  Reuse_cnt  Date
------------ -------- ---------- ------- -------- --------- -------
number of alarms     29    11        27.50    11      -1  Dec 4 2002 10:20AM
(1 row affected)


sp_monitorconfig "number of devices", sample_table

---------------------
(return status = 0)

select * from sample_table

-----------------
Name          Num_free  Num_active  Pct_act  MaxUsed  Reuse_cnt  Date
------------  ---------  ----------  -------  -------   --------   -------
number of alarms  29        11        27.50    11      -1 Dec 4 2002 10:20AM
number of devices 9         1         10.00    1       -1 Dec 4 2002 10:20AM
(2 rows affected)
```

The result set saved to the table accumulates until you delete or truncate the table.

Note If sample_table is in another database, you must provide its fully qualified name in quotes.

Usage

- sp_monitorconfig displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases, such as the number of metadata descriptors currently in use by the server.
- sp_monitorconfig also reports the number of auxiliary scan descriptors in use. A scan descriptor manages a single scan of a table when queries are run on the table.
- sp_monitorconfig monitors the following resources:

  additional network memory
  audit queue size
  heap memory per user
  max cis remote connection
  max memory
  max number network listeners
  memory per worker process
  max online engines
  number of alarms
  number of aux scan descriptors
  number of devices
  number of dtx participants
  number of java sockets
  number of large i/o buffers
  number of locks
  number of mailboxes
  number of messages
  number of open databases
  number of open indexes
  number of open objects
  number of remote connections
  number of remote logins
  number of remote sites
  number of sort buffers
  number of user connections
  number of worker processes
  partition groups
  permission cache entries
  procedure cache size
  size of global fixed heap
  size of process object heap
  size of shared class heap
  size of unilib cache
  txn to pss ratio

- The columns in the sp_monitorconfig output provide the following information:
  - num_free – specifies the number of available metadata or auxiliary scan descriptors not currently used.
  - num_active – specifies the number of metadata or auxiliary scan descriptors installed in cache (that is, active).
  - pct_active – specifies the percentage of cached or active metadata or auxiliary scan descriptors.
  - Max_Used – specifies the maximum number of metadata or auxiliary scan descriptors that have been in use since the server was started.
  - Reused – specifies whether a metadata descriptor was reused in order to accommodate an increase in indexes, objects, or databases in the server. The returned value is `Yes`, `No` or `NA` (for configuration parameters that do not support the reuse mechanism, such as the number of aux scan descriptors).
- Use the value in the Max_Used column as a basis for determining an appropriate number of descriptors; be sure to add about 10 percent for the final setting. For example, if the maximum number of index metadata descriptors used is 142, you might set the number of open indexes configuration parameter to 157.
- If the Reused column states `Yes`, reset the configuration parameter to a higher value. When descriptors need to be reused, there can be performance problems, particularly with open databases. An open database contains a substantial amount of metadata information, which means that to fill up an open database, Adaptive Server needs to access the metadata on the disk many times; the server can also have a spinlock contention problem. To check for spinlock contention, use the system procedure sp_sysmon. For more information, see the *Performance and Tuning Guide*. To find the current number of indexes, objects, or databases, use sp_countmetadata.
- To get an accurate reading, run sp_monitorconfig during a normal Adaptive Server peak time period. You can run sp_monitorconfig several times during the peak period to ensure that you are actually finding the maximum number of descriptors used.
- *result_tbl_name* creates a table using the following syntax. All the result information is saved in this table, which returns no standard output.

```
create table table_name(
    Name varchar(35), Num_free int,
    Num_active int, Pct_act char(6),
    Max_Used int, Reuse_cnt int,
    Date varchar(30))
```

- Some configuration parameter, such as *number of sort buffers* and *txn to pss ratio*, are dependent on the number of configured user connections, while other configuration parameters, such as *max number of network listeners*, are per engine.
- The output of sp_monitorconfig uses the number of user connections and online engines to calculate the values for the columns num_free, num_active, pct_act and max_used.
- The updates on the internal monitor counters are done without using synchronization methods because of performance reasons. For this reason, a multi-engine Adaptive Server under heavy load might report numbers in the sp_monitorconfig output that are not a completely accurate.

Permissions

Only a System Administrator can execute sp_monitorconfig.

See also

System procedures sp_configure, sp_countmetadata, sp_helpconfig, sp_helpconstraint, sp_sysmon

sp_object_stats

Description

Shows lock contention, lock wait-time, and deadlock statistics for tables and indexes.

Syntax

sp_object_stats *interval* [, *top_n*
    [, *dbname*, *objname* [, *rpt_option*]]]

Parameters

interval
specifies the time period for the sample. It must be in HH:MM:SS form, for example "00:20:00".

top_n
the number of objects to report, in order of contention. The default is 10.

dbname
the name of the database to report on. If no database name is given, contention on objects in all databases is reported.

objname
the name of a table to report on. If a table name is specified, the database name must also be specified.

rpt_option
must be either rpt_locks or rpt_objlist.

Examples

Example 1 Reports lock statistics on the top 10 objects server-wide:

```
sp_object_stats "00:20:00"
```

Example 2 Reports only on tables in the pubtune database, and lists the five tables that experienced the highest contention:

```
sp_object_stats "00:20:00", 5, pubtune
```

Example 3 Shows only the names of the tables that had the highest locking activity, even if contention and deadlocking does not take place:

```
sp_object_stats "00:15:00", @rpt_option = "rpt_objlist"
```

Usage

- sp_object_stats reports on the shared, update, and exclusive locks acquired on tables during a specified sample period. The following reports shows the titles tables:

```
Object Name: pubtune..titles (dbid=7,
objid=208003772,lockscheme=Datapages)


 Page Locks     SH_PAGE                UP_PAGE               EX_PAGE$
 ----------    ----------             ----------            ----------
 Grants:          94488                  4052                  4828
```

```
  Waits:               532                     500                     776
  Deadlocks:             4                       0                      24
 Wait-time:      20603764 ms             14265708 ms             2831556 ms

Contention:          0.56%                  10.98%                  13.79%

 *** Consider altering pubtune..titles to Datarows locking.
```

- Table 1-17 shows the meaning of the values.

Table 1-17: Output of sp_object_stats

Output row	Value
Grants	The number of times the lock was granted immediately.
Waits	The number of times the task needing a lock had to wait.
Deadlocks	The number of deadlocks that occurred.
Wait-times	The total number of milliseconds that all tasks spent waiting for a lock.
Contention	The percentage of times that a task had to wait or encountered a deadlock.

- sp_object_stats recommends changing the locking scheme when total contention on a table is more than 15 percent, as follows:
  - If the table uses allpages locking, it recommends changing to datapages locking.
  - If the table uses datapages locking, it recommends changing to datarows locking.
- *rpt_option* specifies the report type:
  - rpt_locks reports grants, waits, deadlocks and wait times for the tables with the highest contention. rpt_locks is the default.
  - rpt_objlist reports only the names of the objects that had the highest level of lock activity.
- sp_object_stats creates a table named tempdb..syslkstats. This table is not dropped when the stored procedure completes, so it can be queried by a System Administrator using Transact-SQL.
- Only one user at a time should execute sp_object_stats. If more than one user tries to run sp_object_stats simultaneously, the second command may be blocked, or the results may be invalid.
- The tempdb..syslkstats table is dropped and re-created each time sp_object_stats is executed.
- The structure of tempdb..syslkstats is described in Table 1-18.

Table 1-18: Columns in the tempdb..syslkstats table

Column name	Datatype	Description
dbid	smallint	Database ID
objid	int	Object ID
lockscheme	smallint	Integer values 1–3: Allpages = 1, Datapages = 2, Datarows = 3
page_type	smallint	Data page = 0, or index page = 1
stat_name	char(30)	The statistics represented by this row
stat_value	float	The number of grants, waits or deadlocks, or the total wait time

The values in the stat_name column are composed of three parts:

- The first part is “ex” for exclusive lock, “sh” for shared lock, or “up” for update lock.
- The second part is “pg” for page locks, or “row” for row locks.
- The third part is “grants” for locks granted immediately, “waits” for locks that had to wait for other locks to be released, “deadlocks” for deadlocks, and “waittime” for the time waited to acquire the lock.

- If you specify a table name, sp_object_stats displays all tables by that name. If more than one user owns a table with the specified name, output for these tables displays the object ID, but not the owner name.

Permissions

Only a System Administrator can execute sp_object_stats.

See also

Commands alter table

sp_passthru

Description

Component Integration Services only Allows the user to pass a SQL command buffer to a remote server.

Syntax

sp_passthru *server*, *command*, *errcode*, *errmsg*, *rowcount*
[, *arg1*, *arg2*, ... *argn*]

Parameters

server
is the name of a remote server to which the SQL command buffer will be passed. The class of this server must be a supported, non-local server class.

command
is the SQL command buffer. It can hold up to 255 characters.

errcode
is the error code returned by the remote server, if any. If no error occurred at the remote server, the value returned is 0.

errmsg
is the error message returned by the remote server. It can hold up to 255 characters. This parameter is set only if *errcode* is a nonzero number; otherwise NULL is returned.

rowcount
is the number of rows affected by the last command in the command buffer. If the command was an insert, delete, or update, this value represents the number of rows affected even though none were returned. If the last command was a query, this value represents the number of rows returned from the external server.

arg1 … argn
receives the results from the last row returned by the last command in the command buffer. You can specify up to 250 *arg* parameters. All must be declared as output parameters.

Examples

Returns the date from the Oracle server in the output parameter *@oradate*. If an Oracle error occurs, the error code is placed in *@errcode* and the corresponding message is placed in *@errmsg*. The *@rowcount* parameter will be set to 1:

```
sp_passthru ORACLE, "select date from dual", @errcode output,
    @errmsg output, @rowcount output, @oradate output
```

Usage

- sp_passthru allows the user to pass a SQL command buffer to a remote server. The syntax of the SQL statement or statements being passed is assumed to be the syntax native to the class of server receiving the buffer. No translation or interpretation is performed. Results from the remote server are optionally placed in output parameters.

  Use sp_passthru only when Component Integration Services is installed and configured.

- You can include multiple commands in the command buffer. For some server classes, the commands must be separated by semicolons. See the *Component Integration Services User's Guide* for a more complete discussion of query buffer handling in passthru mode.

Return Parameters

- The output parameters *arg1* ... *argn* will be set to the values of corresponding columns from the last row returned by the last command in the command buffer. The position of the parameter determines which column's value the parameter will contain. *arg1* receives values from column 1, *arg2* receives values from column 2, and so on.
- If there are fewer optional parameters than there are returned columns, the excess columns are ignored. If there are more parameters than columns, the remaining parameters are set to NULL.
- An attempt is made to convert each column to the datatype of the output parameter. If the datatypes are similar enough to permit *implicit* conversion, the attempt will succeed. For information on implicit conversion, see "Datatype conversion functions" on page 58 in Chapter 2, "Transact-SQL Functions" of *Reference Manual: Building Blocks*. See the *Component Integration Services User's Guide* for information on which datatype represents the datatypes from each server class when in passthru mode.

Permissions

Any user can execute sp_passthru.

See also

System procedures sp_autoconnect, sp_remotesql

sp_password

Description

Adds or changes a password for an Adaptive Server login account.

Syntax

sp_password *caller_passwd*, *new_passwd* [, *loginame*, *@immediate*]

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 *loginame*. 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.

loginame
the login name of the user whose account password is being changed by the System Security Officer.

@immediate
specifies whether a password immediately takes effect on users who are logged in. The values are:

- 0 – users who are logged in keep their old passwords until they reconnect
- 1 – the password changes immediately in the syslogins table, and users who are logged in get their passwords updated while they are still logged in.

Examples

Example 1 Changes your password from password from "3blindmice" to "2mediumhot." (Enclose the passwords in quotes because they begin with numerals.):

```
sp_password "3blindmice", "2mediumhot"
```

Example 2 A System Security Officer whose password is "2tomato" has changed Victoria's password to "sesame1":

```
sp_password "2tomato", sesame1, victoria
```

Example 3 Changes your password from NULL to "16tons." Notice that NULL is not enclosed in quotes (NULL is not a permissible new password):

```
sp_password null, "16tons"
```

Example 4

```
PRODUCTION...sp_password figaro, lilacs
```

Changes your password on the PRODUCTION server from “figaro” to “lilacs.”

Usage

- Any user can change his or her password with sp_password.
- New passwords must be at least 6 characters 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 systemwide password expiration configuration parameter to establish a password expiration interval that forces all Adaptive Server login accounts to change passwords on a regular basis. See the *System Administration Guide* for more information.

Permissions

Only a System Security Officer can execute sp_password to change another user’s password. Any user can execute sp_password to change his or her own password.

See also

System procedures sp_addlogin, sp_adduser

sp_placeobject

Description

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*. Specify index names in the form "*tablename.indexname*"

Examples

Example 1 This places all subsequent space allocation for the table authors on the segment named "segment3":

```
sp_placeobject segment3, authors
```

Example 2 This command places all subsequent space allocation for the employee table's index named employee_nc on the segment named indexes:

```
sp_placeobject indexes, 'employee.employee_nc'
```

Usage

- You cannot change the location of future space allocations for system tables.
- Placing a table or an index on a particular segment does not affect the location of any existing table or index data. It affects only future space allocation. Changing the segment used by a table or an index can spread the data among multiple segments.
- If you use sp_placeobject with a clustered index, the table moves with the index.
- You can specify a segment when you create a table or an 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 an index across more than one disk fragment, the diagnostic command dbcc displays messages about the data that resides on the fragments that were in use for storage before sp_placeobject executed. Ignore those messages.
- You cannot use sp_placeobject on a partitioned table.

Permissions

Only the table owner, Database Owner, or System Administrator can execute sp_placeobject.

See also

Commands alter table, dbcc

System procedures sp_addsegment, sp_dropsegment, sp_extendsegment, sp_helpindex, sp_helpsegment

sp_plan_dbccdb

Description

Recommends suitable sizes for new dbccdb and dbccalt databases, lists suitable devices for dbccdb and dbccalt, and suggests a cache size and a suitable number of worker processes for the target database.

Syntax

sp_plan_dbccdb [*dbname*]

Parameters

dbname
specifies the name of the target database. If *dbname* is not specified, sp_plan_dbccdb makes recommendations for all databases in master..sysdatabases.

Examples

Example 1 Returns configuration recommendations for creating a dbccdb database suitable for checking the master database. The dbccdb database already existed at the time this command was run, so the size of the existing database is provided for comparison:

```
sp_plan_dbccdb master

Recommended size for dbccdb is 4MB.
dbccdb database already exists with size 8MB.
Recommended values for workspace size, cache size and process count are:
dbname           scan ws    text ws    cache    process count
master           64K        64K        640K     1
```

Example 2 Returns configuration recommendations for creating a dbccdb database suitable for checking all databases in the server. No dbccdb database existed at the time this command was run:

```
sp_plan_dbccdb

Recommended minimum size for dbccdb is 4MB.
Recommended values for workspace size, cache size and process count are:
dbname           scan ws    text ws    cache    process count
master           64K        64K        640K     1
tempdb           64K        64K        640K     1
model            64K        64K        640K     1
sybsystemprocs  272K        80K        640K     1
dbccdb          128K        64K        640K     1
```

Example 3 Returns configuration recommendations for creating a dbccdb database suitable for checking pubs2:

```
sp_plan_dbccdb pubs2

Recommended size for dbccdb is 4MB.
Recommended devices for dbccdb are:
```

```
Logical Device Name      Device Size Physical Device Name
sprocdev                 28672      /remote/sybase/devices/srv_sprocs_dat
tun_dat                  8192       /remote/sybase/devices/srv_tun_dat
tun_log                  4096       /remote/sybase/devices/srv_tun_log
Recommended values for workspace size, cache size and process count are:
dbname     scan ws    text ws    cache    process count
pubs2      64K        64K        640K     1
```

Usage

- sp_plan_dbccdb recommends suitable sizes for creating new dbccdb and dbccalt databases, lists suitable devices for the new database, and suggests cache size and a suitable number of worker processes for the target database.
- If you specify dbccdb, sp_plan_dbccdb recommends values for dbccalt, the alternate database. If you specify dbccalt, sp_plan_dbccdb recommends values for dbccdb.
- sp_plan_dbccdb does not report values for existing dbccdb and dbccalt databases. To gather configuration parameters for an existing dbccdb or dbccalt database, use sp_dbcc_evaluatedb.
- For information on the dbcc stored procedures for maintaining dbccdb and for generating reports from dbccdb, see Chapter 4, "dbcc Stored Procedures."

Permissions

Only the System Administrator or Database Owner can execute sp_plan_dbccdb. Only the System Administrator can execute sp_plan_dbccdb without specifying a database name.

See also

Commands dbcc

System procedures sp_dbcc_evaluatedb

sp_poolconfig

Description

Creates, drops, resizes, and provides information about memory pools within data caches.

Syntax

To create a memory pool in an existing cache, or to change pool size:

```
sp_poolconfig cache_name [, "mem_size [P|K|M|G]", "config_poolK"
    [, "affected_poolK"]]
```

To change a pool's wash size:

```
sp_poolconfig cache_name, "io_size ", "wash=size[P|K|M|G]"
```

To change a pool's asynchronous prefetch percentage:

```
sp_poolconfig cache_name, "io_size ",
    "local async prefetch limit=percent "
```

Parameters

cache_name
is the name of an existing data cache.

mem_size
is the size of the memory pool to be created or the new total size for an existing pool, if a pool already exists with the specified I/O size. The minimum size of a pool is 512K. Specify size units with P for pages, K for kilobytes, M for megabytes, or G for gigabytes. The default is kilobytes.

config_pool
is the I/O size performed in the memory pool where the memory is to be allocated or removed.

Valid I/O sizes are multiples of the logical page size, up to four times the amount.

affected_pool
is the size of I/O performed in the memory pool where the memory is to be deallocated. If *affected_pool* is not specified, the memory is taken from the logical page size memory pool.

io_size
is the size of I/O performed in the memory pool where the wash size is to be reconfigured. The combination of cache name and I/O size uniquely identifies a memory pool.

wash=*size*
Changes the wash size (the point in the cache at which Adaptive Server writes dirty pages to disk) for a memory pool.

local async prefetch limit=*percent*
sets the percentage of buffers in the pool that can be used to hold buffers that have been read into cache by asynchronous prefetch, but that have not yet been used.

Examples

Example 1 Creates a 16K pool in the data cache pub_cache with 10MB of space. All space is taken from the default 2K memory pool:

```
sp_poolconfig pub_cache, "10M", "16K"
```

Example 2 Moves 16MB of space to the 32K pool from the 64K pool of pub_cache:

```
sp_poolconfig pub_cache, "16M", "32K", "64K"
```

Example 3 Reports the current configuration of pub_cache:

```
sp_poolconfig "pub_cache"
```

Example 4 Removes the 16K memory pool from pub_cache, placing all of the memory assigned to it in the 2K pool:

```
sp_poolconfig pub_cache, "0K", "16K"
```

Example 5 Changes the wash size of the 2K pool in pubs_cache to 508K:

```
sp_poolconfig pub_cache, "2K", "wash=508K"
```

Example 6 Changes the asynchronous prefetch limit for the 2K pool to 15 percent:

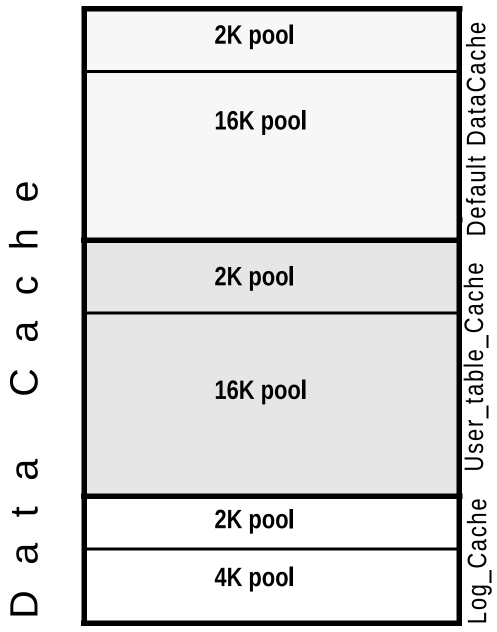
```
sp_poolconfig pub_cache, "2K", "local async prefetch limit=15"
```

Usage

- When you create a data cache with sp_cacheconfig, all space is allocated to the logical page size memory pool. sp_poolconfig divides the data cache into additional pools with larger I/O sizes.
- If no large I/O memory pools exist in a cache, Adaptive Server performs I/O in 2K units, the size of a data page, for all of the objects bound to the cache. You can often enhance performance by configuring pools that perform large I/O. A 16K memory pool reads and writes eight data pages in a single I/O operation.
- The combination of cache name and I/O size must be unique. In other words, you can have only one pool of a given I/O size in a particular data cache.
- Only one sp_poolconfig command can be active on a single cache at one time. If a second sp_poolconfig command is issued before the first one completes, it sleeps until the first command completes.
- Figure 1-3 shows a data cache on a server that uses 2K logical pages with:

- The default data cache with a 2K pool and a 16K pool
- A user cache with a 2K pool and a 16K pool
- A log cache with a 2K pool and a 4K pool

Figure 1-3: Data cache with default and user-defined caches



- You can create pools with I/O sizes up to 16K in the default data cache.
- The minimum size of a memory pool is 512K. You cannot reduce the size of any memory pool in any cache to less than 512K by transferring memory to another pool.
- Two circumstances can create pool less than 512K:
  - If you attempt to delete a pool by setting its size to zero, and some of the pages are in use, sp_poolconfig reduces the pool size as much as possible, and prints a warning message. The status for the pool is set to “Unavailable/deleted”.

    - If you attempt to move buffers to create a new pool, and enough buffers cannot be moved to the new pool, sp_poolconfig moves as many buffers as it can, and the cache status is set to “Unavailable/too small.”

  In both of these cases, you can retry to command at a later time. The pool will also be deleted or be changed to the desired size when the server is restarted.

- You can create memory pools while Adaptive Server is active; no restart is needed for them to take effect. However, Adaptive Server can move only “free” buffers (buffers that are not in use or that do not contain changes that have not been written to disk). When you configure a pool or change its size, Adaptive Server moves as much memory as possible to the pool and prints an informational message showing the requested size and the actual size of the pool. After a restart of Adaptive Server, all pools are created at the configured size.
- The following commands perform only 2K I/O: create database, alter database, some dbcc commands, disk init, and drop table. dbcc checktable can perform large I/O, and dbcc checkdb performs large I/O on tables and 2K I/O on indexes. Also, recovery uses only the 2K memory pool: all pages are read into and changed in the 2K pool of the default cache. Be sure that your default 2K pool is large enough for these activities.
- Most Adaptive Servers perform best with I/O configured for transactions logs that is twice the logical page size. Adaptive Server uses the default I/O size of twice the logical page size if the default cache or a cache with a transaction log bound to it is configured with a memory pool twice the logical page size. Otherwise, it uses the logical page size memory pool.
- You can increase the default log I/O size for a database using the sp_logiosize system procedure. However, the I/O size you specify must have memory pools of the same size in the cache bound to the transaction log. If not, Adaptive Server uses the logical page size memory pools.

Wash percentage

- The default value for the wash size is computed as follows:
    - If the pool size is less than 300MB, the default wash size is set to 20 percent of the buffers in the pool
    - If the pool size is greater than 300MB, the default wash size is 20 percent of the number of buffers in 300MB
- The minimum setting for the wash size is 10 buffers, and the maximum setting is 80 percent of the size of the pool.

- Each memory pool contains a wash area at the least recently used (LRU) end of the chain of buffers in that pool. Once dirty pages (pages that have been changed while in cache) move into the wash area, Adaptive Server initiates asynchronous writes on these pages. The wash area must be large enough so that pages can be written to disk before they reach the LRU end of the pool. Performance suffers when Adaptive Server needs to wait for clean buffers.

  The default percentage, placing 20 percent of the buffers in the wash area, is sufficient for most applications. If you are using an extremely large memory pool, and your applications have a very high data modification rate, you may want to increase the size to 1 or 2 percent of the pool. Contact Sybase Technical Support for more information about choosing an effective wash size.

Local asynchronous prefetch percentage

- The default value for a pool's asynchronous prefetch percentage is set by the configuration parameter global async prefetch limit. The pool limit always overrides the global limit.
- To disable prefetch in a pool (if the global limit is a nonzero number), set the pool's limit to 0.
- See the *Performance and Tuning Guide* for information on the performance impact of changes to the asynchronous prefetch limit.

Permissions

Only a System Administrator can execute sp_poolconfig to reconfigure memory pools within data caches. Any user can use sp_poolconfig to get information about memory pools.

See also

System procedures sp_cacheconfig, sp_helpcache, sp_logiosize, sp_unbindcache, sp_unbindcache_all

sp_primarykey

Description

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 from one to eight columns.

Examples

Example 1 Defines the au_id field as the primary key of the table authors:

```
sp_primarykey authors, au_id
```

Example 2 Defines the combination of the fields lastname and firstname as the primary key of the table employees:

```
sp_primarykey employees, lastname, firstname
```

Usage

- Executing sp_primarykey adds the key to the syskeys table. Only the owner of a table or view can define its primary key. sp_primarykey does not enforce referential integrity constraints; use the primary key clause of the create table or alter table command to enforce a primary key relationship.
- Define keys with sp_primarykey, sp_commonkey, and sp_foreignkey 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.

Permissions

Only the owner of the specified table or view can execute sp_primarykey.

See also

Commands alter table, create table, create trigger

System procedures sp_commonkey, sp_dropkey, sp_foreignkey, sp_helpjoins, sp_helpkey

sp_processmail

Description

Windows NT only Reads, processes, sends, and deletes messages in the Adaptive Server message inbox, using the xp_findnextmsg, xp_readmail, xp_sendmail, and xp_deletemail system extended stored procedures (ESPs).

Syntax

sp_processmail [*subject*] [, *originator* [, *dbuser*
    [, *dbname* [, *filetype* [, *separator*]]]]]

Parameters

subject

is the subject header of the message. If you specify a *subject* but not an *originator*, sp_processmail processes all unread messages in the inbox that has the specified subject header. If you specify both *subject* and *originator*, sp_processmail processes all unread messages with the specified subject header sent by the specified originator. If you do not specify either *subject* or *originator*, sp_processmail processes all the unread messages in the Adaptive Server message inbox.

originator

is the sender of an incoming message. If you specify an *originator* and do not specify a *subject*, sp_processmail processes all unread messages in the inbox sent by the specified originator.

dbuser

specifies the Adaptive Server login name to use for the user context for executing the query in the message. The default is “guest.”

dbname

specifies the database name to use for the database context for executing the query in the message. The default is “master.”

filetype

specifies the file extension of the attached file that contains the results of the query. The default is “.txt”.

separator

specifies the character to use as a column separator in the query results. It is the same as the /s option of isql. The default is the tab character.

Examples

Example 1 Processes all unread messages in the Adaptive Server inbox with the subject header “SQL Report” submitted by mail user “janet”, processes the received queries in the salesdb database as user “sa”, and returns the query results to “janet” in a *.res* file attached to the mail message. The columns in the returned results are separated by semicolons:

```
sp_processmail @subject="SQL REPORT", @originator="janet", @dbuser="sa",
    @dbname="salesdb", @filetype="res", @separator=";"
```

Example 2 Processes all unread messages in the Adaptive Server inbox as user "sa" in the master database and returns the query results in *.txt* files, which are attached to the mail messages. The columns in the returned results are separated by tab characters:

```
sp_processmail @dbuser="sa"
```

Usage

- sp_processmail reads, processes, sends, and deletes messages in the Adaptive Server message inbox, using the xp_findnextmsg, xp_readmail, xp_sendmail, and xp_deletemail system ESPs.
- sp_processmail sends outgoing mail to the originator of the incoming mail message being processed.
- sp_processmail uses the default parameters when invoking the ESPs, except for the *dbuser*, *dbname, attachname,* and *separator* parameters to xp_sendmail, which can be overridden by the parameters to sp_processmail.
- sp_processmail processes all messages as Adaptive Server queries. It reads messages from the Adaptive Server inbox and returns query results to the sender of the message and all its cc'd and bcc'd recipients in an attachment to an Adaptive Server message. sp_processmail generates a name for the attached file consisting of "syb" followed by five random digits, followed by the extension specified by the *filetype* parameter; for example, "syb84840.txt."
- sp_processmail deletes messages from the inbox after processing them.
- The *subject* and *originator* parameters specify which messages should be processed. If neither of these parameters is supplied, sp_processmail processes all the unread messages in the Adaptive Server message inbox.
- sp_processmail does not process attachments to incoming mail. The query must be in the body of the incoming message.

Permissions

Only a System Administrator can execute sp_processmail.

See also

Extended stored procedures xp_deletemail, xp_findnextmsg, xp_readmail, xp_sendmail, xp_startmail

Utility isql

sp_procxmode

Description

Displays or changes the transaction modes associated with stored procedures.

Syntax

sp_procxmode [*procname* [, *tranmode*]]

Parameters

procname
is the name of the stored procedure whose transaction mode you are examining or changing.

tranmode
is the new transaction mode for the stored procedure. Values are "chained", "unchained", and "anymode".

Examples

Example 1 Displays the transaction mode for all stored procedures in the current database:

```
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
```

Example 2 Displays the transaction mode of the stored procedure byroyalty:

```
sp_procxmode byroyalty

procedure name                  transaction mode
------------------------------  ----------------
byroyalty                       Unchained
```

Example 3 Changes the transaction mode for the stored procedure byroyalty in the pubs2 database from “unchained” to “chained”:

```
sp_procxmode byroyalty, "chained"
```

Usage

- To change the transaction mode of a stored procedure, you must be the owner of the stored procedure, the owner of the database containing the stored procedure, or the System Administrator. 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 procname
  ```

- To change a stored procedure's transaction mode, enter:

  ```
  sp_procxmode procname, tranmode
  ```

- When you create a stored procedure, Adaptive Server tags it with the current session's transaction mode. This means:
  - You can execute "chained" stored procedures only in sessions using chained transaction mode.
  - 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, Adaptive Server returns a warning message, but the current transaction, if any, is not affected.

Permissions

Only a System Administrator, the Database Owner, or the owner of a procedure can execute sp_procxmode to change the transaction mode. Any user can execute sp_procxmode to display the transaction mode.

See also

Commands begin transaction, commit, save transaction, set

sp_recompile

Description

Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.

Syntax

sp_recompile *objname*

Parameters

objname
is the name of a table in the current database.

Examples

Recompiles each trigger and stored procedure that uses the table titles the next time the trigger or stored procedure is run:

```
sp_recompile titles
```

Usage

- 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 maximum efficiency.
- sp_recompile looks for *objname* only in the current database and recompiles triggers and stored procedures only in the current database. sp_recompile does not affect objects in other databases that depend on the table.
- You cannot use sp_recompile on system tables.
- In Adaptive Server versions 12.5 and earlier, sp_recompile could influence adhoc queries that you execute. Adaptive Server would return a schema change error (error number 540), and abort the adhoc query. sp_recompile no longer affects such adhoc queries, and you no longer see error 540.

Note sp_recompile could still influence adhoc queries that started execution before sp_recompile was run (a concurrent execution).

Permissions

Any user can execute sp_recompile.

See also

Commands create index, update statistics

sp_remap

Description

Remaps a stored procedure, trigger, rule, default, or view from releases later than 4.8 and prior to 10.0 to be compatible with releases 10.0 and later. Use sp_remap on pre-existing objects that the upgrade procedure failed to remap.

Syntax

sp_remap *objname*

Parameters

objname
is the name of a stored procedure, trigger, rule, default, or view in the current database.

Examples

Example 1 Remaps a stored procedure called myproc:

```
sp_remap myproc
```

Example 2 Remaps a rule called default_date. Execute a use my_db statement to open the my_db database before running this procedure:

```
sp_remap "my_db..default_date"
```

Usage

- If sp_remap fails to remap an object, drop the object from the database and re-create 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. See the *Utility Guide* for more information about defncopy.
- 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.
- You can use sp_remap only on objects in the current database.
- sp_remap makes no changes to objects that were successfully upgraded to the current release.

Permissions

Only a System Administrator or the owner of an object can execute sp_remap.

See also

Commands dump transaction

System procedures sp_helptext

Utility programs defncopy

sp_remoteoption

Description

Displays or changes remote login options.

Syntax

sp_remoteoption [*remoteserver* [, *loginame*
    [, *remotename* [, *optname* [, *optvalue*]]]]]

Parameters

remoteserver
is the name of the server that will be executing RPCs on this server.

Note This manual page uses the term "local server" to refer to the server that is executing the remote procedures that are run from a "remote server."

---

loginame
is the login name that identifies the local login for the *remoteserver*, *loginame*, *remotename* combination.

remotename
is the remote user name that identifies the remote login for the *remoteserver*, *loginame*, *remotename* combination.

optname
is the name of the option to change. 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. Adaptive Server understands any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

optvalue
is either true or false. true turns the option on, false turns it off.

Examples

Example 1 Displays a list of the remote login options:

```
sp_remoteoption
```

```
Settable remote login options.
remotelogin_option
------------------------
trusted
```

Example 2 Defines the remote login from the remote server GATEWAY to be trusted; that is, the password is not checked:

```
sp_remoteoption GATEWAY, churchy, pogo, trusted, true
```

Example 3 Defines the remote login "pogo" from the remote server GATEWAY as a login that is not trusted; that is, the password is checked:

```
sp_remoteoption GATEWAY, churchy, pogo, trusted, false
```

Example 4 Defines all logins from GATEWAY that map to login “albert” on the local server to be trusted:

```
sp_remoteoption GATEWAY, albert, NULL, trusted, true
```

Usage

- To display a list of the remote login options, execute sp_remoteoption with no parameters.
- If you have used sp_addremotelogin to map all users from a remote server to the same local name, specify trusted for those users. For example, if all users from server GOODSRV that are mapped to “albert” are trusted, specify:

```
sp_remoteoption GOODSRV, albert, NULL, trusted, true
```

If the logins are not specified as trusted, they cannot execute RPCs on the local server unless they specify local server passwords when they log into the remote server. When they use Open Client Client-Library, users can specify a password for server-to-server connections with the routine ct_remote_pwd. isql and bcp do not permit users to specify a password for RPC connections.

If users are logged into the remote server using “unified login”, the logins must also be trusted on the local server, or they must specify passwords for the server when they log into the remote server.

See the *System Administration Guide* for more information about setting up servers for remote procedure calls and for using “unified login.”

Permissions

Only a System Security Officer can execute sp_remoteoption.

See also

System procedures sp_addremotelogin, sp_dropremotelogin, sp_helpremotelogin

Utility isql

sp_remotesql

Description

Component Integration Services only Establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client.

Syntax

sp_remotesql *server*, *query*
[, *query2*, ... , *query254*]

Parameters

server_name
is the name of a remote server defined with sp_addserver.

query
is a query buffer a with maximum length of 255 characters.

query2 … query254
is a query buffer with a maximum length of 255 characters. If supplied, these arguments are concatenated with the contents of *query1* into a single query buffer.

Examples

Example 1 Passes the query buffer to FREDS_SERVER, which interprets select @@version and returns the result to the client. Adaptive Server does not interpret the result:

```
sp_remotesql FREDS_SERVER, "select @@version"
```

Example 2 Illustrates the use of sp_remotesql in a stored procedure. This example and example 1 return the same information to the client:

```
create procedure freds_version
as
exec sp_remotesql FREDS_SERVER, "select @@version"
go
exec freds_version
go
```

Example 3 The server concatenates two query buffers into a single buffer, and passes the complete insert statement to the server DCO_SERVER for processing. The syntax for the insert statement is a format that DCO_SERVER understands. The returned information is not interpreted by the server. This example also examines the value returned in *@@error*.

```
sp_remotesql DCO_SERVER,
"insert into remote_table
(numbercol,intcol, floatcol,datecol )",
"values (109.26,75, 100E5,'10-AUG-85')"
select @@error
```

Example 4 Illustrates the use of local variables as parameters to sp_remotesql:

```
declare @servname varchar(30)
declare @querybuf varchar(200)
select @servname = "DCO_SERV"
select @querybuf = "select table_name
    from all_tables
    where owner = 'SYS'"
exec sp_remotesql @servname, @querybuf
```

Usage

- sp_remotesql establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client. The local server does not intercept results.
- You can use sp_remotesql within another stored procedure.
- The query buffer parameters must be a character expression with a maximum length of 255 characters. If you use a query buffer that is not char or varchar, you will receive datatype conversion errors.
- sp_remotesql sets the global variable *@@error* to the value of the last error message returned from the remote server if the severity of the message is greater than 10.
- If sp_remotesql is issued from within a transaction, Adaptive Server verifies that a transaction has been started on the remote server before passing the query buffer for execution. When the transaction terminates, the remote server is directed to commit the transaction. The work performed by the contents of the query buffer is part of the unit of work defined by the transaction.

  If transaction control statements are part of the query buffer, it is the responsibility of the client to ensure that the transaction commit and rollback occur as expected. Mixing Transact-SQL with transaction control commands in the query buffer can cause unpredictable results.
- The local server manages the connection to the remote server. Embedding connect to or disconnect commands in the query buffer causes results that require interpretation by the remote server. This is not required or recommended. Typically, the result is a syntax error.

Permissions

Any user can execute sp_remotesql.

See also

Commands connect to...disconnect

System procedures sp_addserver, sp_autoconnect, sp_passthru

sp_rename

Description

Changes the name of a user-created object or user-defined datatype in the current database.

Syntax

sp_rename *objname*, *newname* [,“*index*” | “*column*”]

Parameters

objname
is the original name of the user-created object (table, view, column, stored procedure, index, trigger, default, rule, check constraint, referential constraint, or user-defined datatype). If the object to be renamed is a column in a table, *objname* must be in the form “*table.column*”. If the object is an index, *objname* must be in the form “*table.indexname*”.

newname
is the new name of the object or datatype. The name must conform to the rules for identifiers and must be unique to the current database.

index
specifies that the object you are renaming is an index, not a column. This argument allows you to rename an index that has the same name as a column, without dropping and re-creating the index.

column
specifies that the object you are renaming is a column, not an index. This argument is part of the same option as the *index* argument.

Examples

Example 1 Renames the titles table to books:

```
sp_rename titles, books
```

Example 2 Renames the title column in the books table to bookname:

```
sp_rename "books.title", bookname
```

Example 3 Renames the titleind index in the books table to titleindex:

```
sp_rename "books.titleind", titleindex
```

Example 4 Renames the user-defined datatype tid to bookid:

```
sp_rename tid, bookid
```

Example 5 renames the title_id index in the titles table to isbn.

```
sp_rename "titles.title_id", isbn, "index"
```

Usage

- sp_rename changes the name of a user-created object or datatype. You can change only the name of an object or datatype in the database in which you issue sp_rename.

- When you are renaming a column or index, do not specify the table name in *newname*. See Examples 2, 3, and 5.
- If a column and an index have the same name, use the [,“*index*” | “*column*”] argument, which specifies whether to rename the index or the column. In the following sample, assume that both an index and a column named idx exist:

  ```
  sp_rename "t.idx", new_idx, "column"
  -------------

  Column name has been changed. (Return status = 0)

  sp_rename "t.idx", new_idx, "index"
  -------------

  Index name has been changed. (Return status = 0)
  ```

- You can change the name of a an object referenced by a view. For example, if a view references the new_sales table and you rename new_sales to old_sales, the view will reference old_sales.
- 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 dropped and re-created. Also, the old object name appears in query results until the user changes and re-creates the procedure, trigger, or view. Change the definitions of any dependent objects when you execute sp_rename. Find dependent objects with sp_depends.

Permissions

Only the Database Owner or a System Administrator can use the setuser command to assume another database user’s identity to rename objects owned by other users. All users can execute sp_rename to rename their own objects.

See also

System procedures sp_depends, sp_rename

sp_renamedb

Description

Changes the name of a user database.

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

Example 1 Renames the accounting database to financial:

```
sp_renamedb accounting, financial
```

Example 2 Renames the database named work, which is a Transact-SQL reserved word, to workdb. This example shows how sp_dboption is used to place the work database in single-user mode before renaming it and restore it to multi-user mode afterward:

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

Usage

- sp_renamedb changes the name of a database. You *cannot* rename system databases or databases with external referential integrity constraints.
- The System Administrator must place a database in single-user mode with sp_dboption before renaming it and must restore it to multi-user mode afterward.

- 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(frgndbid)
  from sysreferences
  where frgndbid 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(pmrydbid)
  from sysreferences
  where pmrydbid is not null
  ```

  Use alter table to drop the cross-database constraints in these tables. Then, rerun sp_renamedb.

- When you change a database name:
  - Drop all stored procedures, triggers, and views that include the database name
  - Change the source text of the dropped objects to reflect the new database name
  - Re-create the dropped objects
  - Change all applications and SQL source scripts that reference the database, either in a use *database_name* command or as part of a fully qualified identifier (in the form *dbname*.[*owner*].*objectname*)
- If you use scripts to run dbcc commands or dump database and dump transaction commands on your databases, be sure to update those scripts.

Warning! Procedures, triggers, and views that depend on a database whose name has been changed work until they are re-created. Change the definitions of any dependent objects when you execute sp_renamedb. Find dependent objects with sp_depends.

Permissions

Only a System Administrator can execute sp_renamedb.

See also

Commands create database

System procedures sp_changedbowner, sp_dboption, sp_depends, sp_helpdb, sp_rename

sp_rename_qpgroup

Description

Renames an abstract plan group.

Syntax

sp_rename_qpgroup *old_name*, *new_name*

Parameters

old_name
is the current name of the abstract plan group.

new_name
is the new name for the group. The specified *new_name* cannot be the name of an existing abstract plan group in the database.

Examples

```
sp_rename_qpgroup dev_plans, prod_plans
```

Changes the name of the group from dev_plans to prod_plans.

Usage

- Use sp_rename_qpgroup to rename an abstract plan group. You cannot use the name of an existing plan group for the new name.
- sp_rename_qpgroup does not affect the contents of the renamed group. IDs of existing abstract plans are not changed.
- You cannot rename the default abstract plan groups, ap_stdin and ap_stdout.
- sp_rename_qpgroup cannot be run in a transaction.

Permissions

Only a System Administrator or the Database Owner can execute sp_rename_qpgroup.

See also

System procedures sp_help_qpgroup

sp_reportstats

Description: Reports statistics on system usage.

Syntax: sp_reportstats [*loginame*]

Parameters: *loginame*
is the login name of the user to show accounting totals for.

Examples: **Example 1** Displays a report of current accounting totals for all Adaptive Server users:

```
sp_reportstats

Name    Since         CPU     Percent CPU  I/O     Percent I/O
------  -----------   -----   ------------ -----   -------------
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%

Total CPU   Total I/O
---------   ---------
40006       20555
```

Example 2 Displays a report of current accounting totals for user “kathy”:

```
sp_reportstats kathy

Name    Since         CPU     Percent CPU   I/O     Percent I/O
------  -----------   -----   ------------  -----   -------------
kathy   Jul 24 1993   498     49.8998%      48392   9.1829%

Total CPU   Total I/O
---------   ----------
998         98392
```

Usage:

- 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. sp_reportstats accepts one parameter, the login name of the account to report. With no parameters, sp_reportstats reports on all accounts.
- sp_reportstats does not report statistics for any process with a system user ID (suid) of 0 or 1. This includes deadlock detection, checkpoint, housekeeper, network, auditing, mirror handlers, and all users with sa_role.

- The units reported for “CPU” are *machine* clock ticks, not Adaptive Server clock ticks.
- The “probe” user exists for the two-phase commit probe process, which uses a challenge-and-response mechanism to access Adaptive Server.

Permissions

Only a System Administrator can execute sp_reportstats.

See also

System procedures sp_clearstats, sp_configure

sp_revokelogin

Description

Windows NT only Revokes Adaptive Server roles and default permissions from Windows NT users and groups when Integrated Security mode or Mixed mode (with Named Pipes) is active.

Syntax

sp_revokelogin {*login_name* | *group_name*}

Parameters

login_name
is the network login name of the Windows NT user.

group_name
is the Windows NT group name.

Examples

Example 1 Revokes all permissions from the Windows NT user named "jeanluc":

```
sp_revokelogin jeanluc
```

Example 2 Revokes all roles from the Windows NT Administrators group:

```
sp_revokelogin Administrators
```

Usage

- Use sp_revokelogin only when Adaptive Server is running in Integrated Security mode or Mixed mode, when the connection is Named Pipes. If Adaptive Server is running in Standard mode, or in Mixed mode using a connection other than Named Pipes, use the revoke command.
- If you revoke a user's roles and default privileges with sp_revokelogin, that user can no longer log into Adaptive Server over a trusted connection.

Permissions

Only a System Administrator can execute sp_revokelogin.

See also

Commands grant, revoke, setuser

System procedures sp_droplogin, sp_dropuser, sp_logininfo

sp_role

Description

Grants or revokes roles to an Adaptive Server login account.

Syntax

sp_role {"grant" | "revoke"}, *rolename*, *loginame*

Parameters

grant | revoke
specifies whether to grant the role to or revoke the role from *loginame*.

rolename
is the role to be granted or revoked.

loginame
is the login account to or from which the role is to be granted or revoked.

Examples

Grants the System Administrator role to the login account named “alexander”:

```
sp_role "grant", sa_role, alexander
```

Usage

- sp_role grants or revokes roles to an Adaptive Server login account.
- When you grant a role to a user, it takes effect the next time the user logs into Adaptive Server. Alternatively, the user can enable the role immediately by using the set role command. For example, the command enables the System Administrator role for the user:

  ```
  set role sa_role on
  ```

- 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 active (on). To turn a role off, use the set command. For example, to deactivate the System Administrator role, use the command:

  ```
  set role "sa_role" off
  ```

Permissions

Only a System Administrator can execute sp_role to grant the System Administrator role to other users. Only a System Security Officer can execute sp_role to grant any role other than “sa” to other users.

See also

Commands grant, revoke, set

Functions proc_role

System procedures sp_displaylogin

sp_sendmsg

Description

Sends a message to a User Datagram Protocol (UDP) port.

Syntax

sp_sendmsg *ip_address*, *port_number*, *message*

Parameters

ip_address
is the IP address of the machine where the UDP application is running.

port_number
is the port number of the UDP port.

message
is the message to send. It can be up to 255 characters in length.

Examples

```
sp_sendmsg "120.10.20.5", 3456, "Hello World"
```

Usage

- sp_sendmsg is not supported on Windows NT.
- To enable the use of UDP messaging, a System Security Officer must set the configuration parameter allow sendmsg to 1.
- No security checks are performed with sp_sendmsg. Sybase strongly recommends caution when using sp_sendmsg to send sensitive information across the network. By enabling this functionality, the user accepts any security problems which result from its use.
- This sample C program listens on a port that you specify and echoes the messages it receives. For example, to receive the sp_sendmsg calls for Example 1, use:

```
updmon 3456
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>

main(argc, argv)
int argc; char *argv[];
{
        struct sockaddr_in sadr;
        int portnum,sck,dummy,msglen;
        char msg[256];

        if (argc < 2) {
                printf(“Usage: udpmon <udp portnum>\n”);
                exit(1);
        }

        if ((portnum=atoi(argv[1])) < 1) {
                printf("Invalid udp portnum\n");
                exit(1);
        }

        if ((sck=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)) < 0) {
                printf("Couldn't create socket\n");
                exit(1);
        }

        sadr.sin_family = AF_INET;
        sadr.sin_addr.s_addr = inet_addr("0.0.0.0");
        sadr.sin_port = portnum;

        if (bind(sck,&sadr,sizeof(sadr)) < 0) {
                printf("Couldn't bind requested udp port\n");
                exit(1);
        }

        for (;;)
        {
                if((msglen=recvfrom(sck,msg,sizeof(msg),0,NULL,&dummy))
< 0)
                        printf("Couldn't recvfrom() from udp port\n");
                printf("%.*s\n", msglen, msg);
        }
}
```

Permissions

Any user can execute sp_sendmsg.

See also

Function syb_sendmsg

sp_serveroption

Description

Displays or changes remote server options.

Syntax

sp_serveroption [*server*, *optname*, *optvalue*]

Parameters

server

is the name of the remote server for which to set the option.

optname

is the name of the option to be set or unset. Table 1-19 lists the option names.

Table 1-19: sp_serveroption options

Option	Meaning
mutual authentication	*Valid for "rpc security model B" only* – this option specifies that the local server authenticates the remote server by retrieving the credential of the remote server and verifying it with the security mechanism. With this service, the credentials of both servers are authenticated and verified.
external engine auto start	Specifies that EJB Server starts up each time Adaptive Server starts up. The default is true; starting Adaptive Server also starts up EJB Server.
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.
readonly	*Component Integration Services only* – specifies that access to the server named is read only.
rpc security model A	The default model for handling RPCs. This model does not support mutual authentication, message integrity, or message confidentiality between the local server and the remote server.
rpc security model B	This model results in a single, secure physical connection established between the local and remote servers. Logical connections for each RPC that is executed are multiplexed over the single, secure, physical connection. This model supports mutual authentication, message confidentiality via encryption, and message integrity.
security mechanism	*Valid for "rpc security model B" only* – this option specifies the security mechanism for the remote server. You must set this option to true to use security model B.
server cost	*Component Integration Services only* – specifies the cost of a single exchange under the user's control, on a per-server basis. See Chapter 2, "Understanding Component Integration Services" in *Understanding CIS* for more information.
server login	*Component Integration Services only* – To fully support remote logins, Client-Library provides connection properties that enable CIS to request a server connection. This connection is recognized at the receiving server as a server connection (as opposed to an ordinary client connection), allowing the remote server to validate the connection through the use of sysremotelogins as if the connection were made by a site handler.

Option	Meaning
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.
use message confidentiality	*Valid for "rpc security model B" only* – this option specifies that messages are encrypted when sent to the remote server, and results from the remote server are encrypted.
use message integrity	*Valid for "rpc security model B" only* – this option specifies that messages between the servers are checked for tampering.

Adaptive Server accepts any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

optvalue

is true (on) or false (off) for all options except the security mechanism option.

For the security mechanism option, specify the name of the security mechanism. To see the names of the security mechanisms available on a server, execute:

```
select * from syssecmechs
```

Examples

Example 1 Displays a list of the server options:

```
sp_serveroption

Settable server options.

 ------------------------
 mutual authentication
 net password encryption
 readonly
 rpc security model A
 rpc security model B
 security mechanism
 timeouts
 use message confidentiality
 use message integrity
 timeouts
 net password encryption
```

Example 2 Tells the server not to time out inactive physical connections with the remote server GATEWAY:

```
sp_serveroption GATEWAY, "timeouts", false
```

Example 3 Specifies that when connecting to the remote server GATEWAY, GATEWAY sends back an encryption key to encrypt the password to send to it:

```
sp_serveroption GATEWAY, "net password encryption", true
```

Example 4 Specifies that the EJB Server SYB_EJB starts up each time Adaptive Server starts up:

```
sp_serveroption SYB_EJB, "external engine auto start", true
```

Example 5 Specifies that the security model for RPCs for the server “TEST3” is security model B.

```
sp_serveroption TEST3, "rpc security model B", true
```

Example 6 Specifies that the security model to use for RPCs for “TEST3” is DCE:

```
sp_serveroption TEST3, "security mechanism", dce
```

Example 7 Specifies that the local server will check the authenticity of the remote server “TEST3”. With security model B, the remote server will check the authenticity of the local server, whether or not this option is set:

```
sp_serveroption TEST3, "mutual authentication", true
```

Example 8 Disables automatic startup, where SYB_EJB is the logical name of the EJB Server:

```
sp_serveroption 'SYB_EJB', 'external engine auto start', 'false'
```

To enable automatic startup, enter:

```
sp_serveroption 'SYB_EJB', 'external engine auto start', 'true'
```

See “Starting EJB Server automatically” in Chapter 2, “Getting Started” of the *EJB Server User’s Guide* for more information about using external engine auto start.

Usage

- To display a list of server options that can be set by the user, use sp_serveroption with no parameters.
- Once timeouts is set to false, the site handlers will 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 initiating a remote procedure call. If net password encryption is 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 a command line option for isql. For more information, see the Utility Programs manual for your platform.
- You cannot use the net password encryption option when connecting to a pre-release 10.0 SQL Server.
- The options security mechanism, mutual authentication, use message confidentiality, and use message integrity do *not* apply to security model A.
- To use security model B, both the local server and the remote server must use model B and both must use the same security mechanism.

Permissions

Only a System Administrator can execute sp_serveroption to set the timeouts option. Any user can execute sp_serveroption with no parameters to display a list of options.

Only a System Security Officer can set the net password encryption, security mechanism, mutual authentication, use message confidentiality, and use message integrity options.

See also

Documents Se the *System Administration Guide* for more information on server options.

System procedures sp_helpserver, sp_password

Utility isql

sp_setlangalias

Description

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

```
sp_setlangalias french, français
```

This command assigns the alias name “français” for the official language name “french”.

Usage

- *alias* replaces the current value of syslanguages.alias for the official name.
- The set language command can use the new *alias* in place of the official language name.

Permissions

Only a System Administrator can execute sp_setlangalias.

See also

Commands set

System procedures sp_addlanguage, sp_droplanguage, sp_helplanguage

sp_setpglockpromote

Description

Sets or changes the lock promotion thresholds for a database, for a table, or for Adaptive Server.

Syntax

sp_setpglockpromote {"database" | "table"}, *objname*, *new_lwm*,
    new_hwm, *new_pct*

sp_setpglockpromote server, NULL, *new_lwm*, *new_hwm*, *new_pct*

Parameters

server
: sets server-wide values for the lock promotion thresholds.

"database" | "table"
: specifies whether to set the lock promotion thresholds for a database or table. “database” and “table” are Transact-SQL keywords, so the quotes are required.

objname
: is either the name of the table or database for which you are setting the lock promotion thresholds or null, if you are setting server-wide values.

new_lwm
: specifies the value to set for the low watermark (LWM) threshold. The LWM must be less than or equal to the high watermark (HWM). The minimum value for LWM is 2. This parameter can be null.

new_hwm
: specifies the value to set for the lock promotion HWM threshold. The HWM must be greater than or equal to the LWM. The maximum HWM is 2,147,483,647. This parameter can be null.

new_pct
: specifies the value to set for the lock promotion percentage (PCT) threshold. PCT must be between 1 and 100. This parameter can be null.

Examples

Example 1 Sets the server-wide lock promotion LWM to 200, the HWM to 300, and the PCT to 50:

```
sp_setpglockpromote "server", NULL, 200, 300, 50
```

Example 2 Sets lock promotion thresholds for the master database:

```
sp_setpglockpromote "database", master, 1000, 1100, 45
```

Example 3 Sets lock promotion thresholds for the titles table in the pubs2 database. This command must be issued from the pubs2 database:

```
sp_setpglockpromote "table", "pubs2..titles", 500, 700, 10
```

Example 4 Changes the HWM threshold to 1600 for the master database. The thresholds were previously set with sp_setpglockpromote. This command must be issued from the master database:

```
sp_setpglockpromote "database", master, @new_hwm=1600
```

Usage

- sp_setpglockpromote configures the lock promotion values for a table, for a database, or for Adaptive Server.

  Adaptive Server acquires page locks on a table until the number of locks exceeds the lock promotion threshold. sp_setpglockpromote changes the lock promotion thresholds for an object, a database, or the server. If Adaptive Server is successful in acquiring a table lock, the page locks are released.

  When the number of locks on a table exceeds the HWM threshold, Adaptive Server attempts to escalate to a table lock. When the number of locks on a table is below the LWM, Adaptive Server does not attempt to escalate to a table lock. When the number of locks on a table is between the HWM and LWM and the number of locks exceeds the PCT threshold, Adaptive Server attempts to escalate to a table lock.
- Lock promotion thresholds for a table override the database or server-wide settings. Lock promotion thresholds for a database override the server-wide settings.
- Lock promotion thresholds for Adaptive Server do not need initialization, but you must initialize database and table lock promotion thresholds by specifying LWM, HWM, and PCT with sp_setpglockpromote, which creates a row for the object in sysattributes when it is first run for a database or table. Once the thresholds have been initialized, then they can be modified individually, as in Example 4.
- For a table or a database, sp_setpglockpromote sets LWM, HWM, and PCT in a single transaction. If sp_setpglockpromote encounters an error while updating any of the values, then all changes are aborted and the transaction is rolled back. For server-wide changes, one or more thresholds may fail to be updated while others are successfully updated. Adaptive Server returns an error message if any values fail to be updated.
- To view the server-wide settings for the lock promotion thresholds, use sp_configure "lock promotion" to see all three threshold values. To view lock promotion settings for a database, use sp_helpdb. To view lock promotion settings for a table, use sp_help.

Permissions

Only a System Administrator can execute sp_setpglockpromote.

See also

System procedures sp_configure, sp_dropglockpromote, sp_help, sp_helpdb

sp_setpsexe

Description

Sets custom execution attributes for a session while the session is active.

Syntax

sp_setpsexe *spid*, *exeattr*, *value*

Parameters

spid
is the ID of the session for which to set execution variables. Use sp_who to see spids.

exeattr
identifies the execution attribute to be set. Values are priority and enginegroup.

value
is the new value of exeattr. Values for each attribute are as follows:

- If *exeattr* is priority, *value* is HIGH, MEDIUM, or LOW.
- If *exeattr* is enginegroup, *value* is the name of an existing engine group.

Examples

This example sets the priority of the process with an ID of 1 to HIGH:

```
sp_setpsexe 1, "priority", "HIGH"
```

Usage

- Execution attribute values specified with sp_setpsexe are valid for the current session only and do not apply after the session terminates.
- Use sp_setpsexe with caution or it can result in degraded performance. Changing attributes "on the fly", using sp_setpsexe, can help if the process is not getting CPU time; however, if the performance problem is due to something else, such as locks, changing execution attributes could make the problem worse.
- Because you can only set execution attributes for sessions, sp_setpsexe cannot be set for a worker process spid.
- Except for the housekeeper spid, you cannot set execution attributes for system spids.
- sp_setpsexe does not work if there are no online engines in the associated engine group.

Permissions

Only a System Administrator can execute sp_setpsexe without restriction. Any user can execute sp_setpsexe to lower the priority of a process owned by that user.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_dropexeclass, sp_showexeclass

sp_set_qplan

Description

Changes the text of the abstract plan of an existing plan without changing the associated query.

Syntax

sp_set_qplan *id*, *plan*

Parameters

id
is the ID of the abstract plan.

plan
is a new abstract plan.

Examples

```
sp_set_qplan 563789159,
        "( g_join (scan t1) (scan t2))"
```

Usage

- Use sp_set_qplan to change the abstract plan of an existing plan. You can specify a maximum of 255 characters for a plan. If the abstract plan is longer than 255 characters, you can drop the old plan with sp_drop_qplan and then use create plan to create a new plan for the query.
- When you change a plan with sp_set_qplan, plans are not checked for valid abstract plan syntax. Also, the plan is not checked for compatibility with the SQL text. All plans modified with sp_set_qplan should be immediately checked for correctness by running the query for the specified ID.
- To find the ID of a plan, use sp_help_qpgroup, sp_help_qplan, or sp_find_qplan. Plan IDs are also returned by create plan and are included in showplan output.

Permissions

Any user can execute sp_set_qplan to change the text for a plan that he or she owns. Only the System Administrator or the Database Owner can change the text for a plan that belongs to another user.

See also

Commands create plan

System procedures sp_drop_qpgroup, sp_drop_qplan, sp_find_qplan, sp_help_qplan

sp_setrowlockpromote

Description

Sets or changes row-lock promotion thresholds for a datarows-locked table, for all datarows-locked tables in a database, or for all datarows-locked tables on a server.

Syntax

sp_setrowlockpromote "server", NULL, *new_lwm*, *new_hwm*, *new_pct*

sp_setrowlockpromote {"database" | "table"}, *objname*, *new_lwm*,
    new_hwm, *new_pct*

Parameters

server
  sets server-wide values for the row lock promotion thresholds.

"database" | "table"
  specifies whether to set the row-lock promotion thresholds for a database or table.

objname
  is either the name of the table or database for which you are setting the row-lock promotion thresholds or null, if you are setting server-wide values.

new_lwm
  specifies the value to set for the low watermark (LWM) threshold. The LWM must be less than or equal to the high watermark (HWM). The minimum value for LWM is 2. This parameter can be null.

new_hwm
  specifies the value to set for the high watermark (HWM) threshold. The HWM must be greater than or equal to the LWM. The maximum HWM is 2,147,483,647. This parameter can be null.

new_pct
  specifies the value to set for the lock promotion percentage (PCT) threshold. PCT must be between 1 and 100. This parameter can be null.

Examples

Example 1 Sets row lock promotion values for all datarows-locked tables in the engdb database:

```
sp_setrowlockpromote "database", engdb, 400, 400,95
```

Example 2 Sets row lock promotion values for the sales table:

```
sp_setrowlockpromote "table", sales, 250, 250, 100
```

Usage

- sp_setrowlockpromote sets or changes row-lock promotion thresholds for a table, a database, or Adaptive Server.

  Adaptive Server acquires row locks on a datarows-locked table until the number of locks exceeds the lock promotion threshold. If Adaptive Server is successful in acquiring a table lock, the row locks are released.

When the number of row locks on a table exceeds the HWM, Adaptive Server attempts to escalate to a table lock. When the number of row locks on a table is below the LWM, Adaptive Server does not attempt to escalate to a table lock. When the number of row locks on a table is between the HWM and LWM, and the number of row locks exceeds the PCT threshold as a percentage of the number of rows in a table, Adaptive Server attempts to escalate to a table lock.

- Lock promotion is always two-tiered, that is, row locks are promoted to table locks. Adaptive Server does not promote from row locks to page locks.
- Lock promotion thresholds for a table override the database or server-wide settings. Lock promotion thresholds for a database override the server-wide settings.
- To change the lock promotion thresholds for a database, you must be using the master database. To change the lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- Server-wide row lock promotion thresholds can also be set with sp_configure. When you use sp_setrowlockpromote to change the values server-wide, it changes the configuration parameters, and saves the configuration file. When you first install Adaptive Server, the server-wide row lock promotion thresholds set by the configuration parameters are:

row lock promotion HWM	200
row lock promotion LWM	200
row lock promotion PCT	100

See the *System Administration Guide* for more information.

- The system procedure sp_sysmon reports on row lock promotions.
- Database-level row lock promotion thresholds are stored in the master..sysattributes table. If you dump a database, and load it only another server, you must set the row lock promotion thresholds on the new server. Object-level row lock promotion thresholds are stored in the sysattributes table in the user database, and are included in the dump.

Permissions

Only a System Administrator can execute sp_setrowlockpromote.

See also

System procedures sp_configure, sp_droprowlockpromote, sp_sysmon

sp_setsuspect_granularity

Description

Displays or sets the recovery fault isolation mode for a user database, which governs how recovery behaves when it detects data corruption.

Syntax

sp_setsuspect_granularity [*dbname*
    [, "database" | "page" [, "read_only"]]]

Parameters

dbname
is the name of the database for which to display or set the recovery fault isolation mode. For displaying, the default is the current database. For setting, you must be in the master database and specify the target *dbname*.

database
marks the entire database suspect, which makes it inaccessible, if the recovery process detects that any of its data is suspect.

page
marks only the corrupt pages suspect, making them inaccessible, if recovery detects corrupt data in the database. The rest of the data is accessible.

read_only
if specified, marks the entire database read only if recovery marks any pages suspect.

Examples

Example 1 Displays the recovery fault isolation mode for the current database:

```
sp_setsuspect_granularity

DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  ------------------  ------------------  -----------
pubs2    database           database           read/write
```

Example 2 Displays the current and configured recovery fault isolation mode for the pubs2 database:

```
sp_setsuspect_granularity pubs2
```

Example 3 The next time recovery runs in the pubs2 database, if any corrupt pages are detected, only the suspect pages will be taken offline and the rest of the database will be brought online:

```
sp_setsuspect_granularity pubs2, "page"

DB Name       Cur. Suspect Gran. Cfg. Suspect Gran.
------------- ------------------ -----------------
pubs2          database             database
sp_setsuspect_granularity: The new values will become effective
    during the next recovery of the database 'pubs2'.
```

Example 4 The next time recovery runs in the pubs2 database, if any corrupt pages are detected, only the suspect pages will be taken offline and the rest of the database will be brought online in read only mode:

```
sp_setsuspect_granularity pubs2, "page", "read_only"
```

Example 5 The next time recovery runs in the pubs2 database, if any corrupt data is detected, the entire database will be marked suspect and taken offline:

```
sp_setsuspect_granularity pubs2, "database"
```

Usage

- sp_setsuspect_granularity displays and sets the recovery fault isolation mode. This mode governs whether recovery marks an entire database or only the corrupt pages suspect when it detects that any data that it requires has been corrupted. Se the *System Administration Guide* for more information.
- The default recovery fault isolation mode of a user database is “database”. You can set the recovery fault isolation mode only for a user database, not for a system database.
- You must be in the master database to set the recovery fault isolation mode.
- Data marked suspect due to corruption persists across Adaptive Server start-ups. When certain pages have been marked suspect, they remain offline after you reboot the server.
- When part or all of a database is marked suspect, the suspect data is not accessible to users unless a System Administrator has made the suspect data accessible with the sp_forceonline_dband sp_forceonline_page procedures.
- General database corruption, such as a corrupt database log or the unavailability of another resource not specific to a page, causes the entire database to be marked suspect, even if the recovery fault isolation mode is “page”.
- If you do not specify page or database, Adaptive Server displays the current and configured settings. The current setting is the one that was in effect the last time recovery was executed in the database. The configured setting is the one that will be in effect the next time recovery is executed in the database.

- If the database comes online in read_only mode, no user can modify any of its data, including data that is unaffected by the suspect pages and is thus online. However, the system administrator can make the database writeable using the sp_dboption system procedure to set read only to false. In this case, users could then modify the online data, but the suspect data would remain inaccessible.

Permissions

Only a System Administrator can execute sp_setsuspect_granularity to set the recovery fault isolation mode. Any user can execute sp_setsuspect_granularity to display the settings.

See also

Commands dump database, dump transaction, load database

System procedures sp_dboption, sp_forceonline_db, sp_forceonline_page, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_threshold

sp_setsuspect_threshold

Description

Displays or sets the maximum number of suspect pages that Adaptive Server allows in a database before marking the entire database suspect.

Syntax

sp_setsuspect_threshold [*dbname* [, *threshold*]]

Parameters

dbname
is the name of the database for which you want to display or set the suspect escalation threshold. The default is the current database.

threshold
indicates the maximum number of suspect data pages that recovery will allow before marking the entire database suspect. The default is 20 pages. The minimum is 0.

Examples

Example 1 Sets the maximum number of suspect pages to 5. If there are more than 5 suspect pages, recovery will mark the entire database suspect:

```
sp_setsuspect_threshold pubs2, 5
```

Example 2 Displays the current and configured settings for the suspect escalation threshold for the pubs2 database:

```
sp_setsuspect_threshold pubs2
```

Example 3 Displays the current and configured settings for the recovery fault isolation threshold for the current user database:

```
sp_setsuspect_threshold
```

Usage

- You must be in the master database to set the suspect escalation threshold with sp_setsuspect_threshold.
- If you do not specify the number of pages, Adaptive Server displays the current and configured settings. The current setting is the one that was in effect the last time recovery was executed in the database. The configured setting is the one that will be in effect the next time recovery is executed in the database.

Permissions

Only a System Administrator can execute sp_setsuspect_threshold to set the escalation threshold. Any user can execute sp_setsuspect_threshold to display the current settings.

See also

System procedures sp_forceonline_db, sp_forceonline_page, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_granularity

sp_showcontrolinfo

Description

Displays information about engine group assignments, bound client applications, logins, and stored procedures.

Syntax

sp_showcontrolinfo [*object_type*, *object_name*, *spid*]

Parameters

object_type
is AP for application, LG for login, PR for stored procedure, EG for engine group, or PS for process. If you do not specify an *object_type* or specify an *object_type* of null, sp_showcontrolinfo displays information about all types.

object_name
is the name of the application, login, stored procedure, or engine group. Do not specify an *object_name* if you specify PS as the *object_type*. If you do not specify an *object_name* (or specify an *object_name* of null), sp_showcontrolinfo displays information about all object names.

spid
is the Adaptive Server process ID. Specify an *spid* only if you specify PS as the *object_type*. If you do not specify an *spid* (or specify an *spid* of null), sp_showcontrolinfo displays information for all spids. Use sp_who to see spids.

Examples

Example 1 Shows all user-assigned execution class-to-object bindings:

```
sp_showcontrolinfo
```

Example 2 Displays the execution class of the isql application:

```
sp_showcontrolinfo 'AP', 'isql'
```

Example 3 Displays the execution class for all processes assigned to engine groups:

```
sp_showcontrolinfo 'PS'
```

Example 4 Displays the execution class for spid 7:

```
sp_showcontrolinfo 'PS', null, 7
```

Usage

- When used with no parameters, sp_showcontrolinfo displays information about all user-assigned engine group assignments, bound client applications, logins, and stored procedures. When used with the object_type parameter, sp_showcontrolinfo provides information on an individual basis about application, login, or stored procedure bindings to an execution class, engine group compositions, and session-level attribute bindings. See the *Performance and Tuning Guide* for more information.

- Unless object_type is PR, execute sp_showcontrolinfo from the master database. If object_type is PR, execute sp_showcontrolinfo from the database in which the procedure resides.
- If *object_type* is null, sp_showcontrolinfo displays execution class information for objects that match the other parameters.
- If *object_name* is null, sp_showcontrolinfo displays the binding information for all applications, logins, and stored procedures.
- If *spid* is null, sp_showcontrolinfo displays execution class information for objects that match the other parameters.

Permissions

Any user can execute sp_showcontrolinfo.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showexeclass, sp_showpsexe, sp_unbindexeclass, sp_who

Utility isql

sp_showexeclass

Description

Displays the execution class attributes and the engines in any engine group associated with the specified execution class.

Syntax

sp_showexeclass [*execlassname*]

Parameters

execlassname
is the name of an execution class.

Examples

Example 1 Displays the priority and engine group attribute values for all execution classes:

```
sp_showexeclass

classname   priority   engine_group   engines
---------  ---------- ------------- -----------
EC1         HIGH       ANYENGINE           ALL
EC2         MEDIUM     ANYENGINE           ALL
EC3         LOW        LASTONLINE          0
```

Example 2 Displays the attribute values of execution class EC1:

```
sp_showexeclass 'EC1'

classname   priority   engine_group            engines
----------- ---------- ---------------------- -----------
EC1         HIGH       ANYENGINE               ALL
```

Usage

- sp_showexeclass displays the execution class attributes and the engines in any engine group associated with *execlassname*. See the *Performance and Tuning Guide* for more information.
- If *execlassname* is NULL or absent, sp_showexeclass displays the priority and engine group attribute values for all execution classes, including the attribute values of the system-defined classes EC1, EC2, and EC3.

Permissions

Any user can execute sp_showexeclass.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_dropexeclass, sp_showcontrolinfo, sp_unbindexeclass

sp_showplan

Description

Displays the showplan output for any user connection for the current SQL statement or for a previous statement in the same batch.

Syntax

sp_showplan *spid*, *batch_id* output, *context_id* output, *stmt_num* output

To display the showplan output for the current SQL statement without specifying the batch_id, context_id, or stmt_num:

sp_showplan *spid*, null, null, null

Parameters

spid
is the process ID for any user connection. Use sp_who to see spids.

batch_id
is a unique, nonnegative number for a batch

context_id
is a unique number for every procedure (or trigger) executed in a batch.

stmt_num
is the number of the current statement within a batch. The *stmt_num* must be a positive number.

Examples

Example 1 Displays the query plan for the current statement running in the user session with a *spid* value of 99, as well as values for the *batch_id*, *context_id*, and *statement_id* parameters. These values can be used to retrieve query plans in subsequent iterations of sp_showplan for the user session with a *spid* of 99:

```
declare @batch int
declare @context int
declare @statement int
exec sp_showplan 99, @batch output, @context output,
@statement output
```

Example 2 Displays the showplan output for the current statement running in the user session with a *spid* value of 99:

```
sp_showplan 99, null, null, null
```

Usage

- sp_showplan displays the showplan output for a currently executing SQL statement or for a previous statement in the same batch.
- To see the query plan for the previous statement within the same batch, execute sp_showplan again with the same parameter values, but subtract 1 from the statement number. Using this method, you can view all the statements in the statement batch back to query number one.

- sp_showplan can be run independently of Adaptive Server Monitor™ Server.
- If the *context_id* is greater than 0 for a SQL batch, the current statement is embedded in a stored procedure (or trigger) called from the original SQL batch. Select the sysprocesses row with the same *spid* value to display the procedure ID and statement ID.

Permissions

Only a System Administrator can execute sp_showplan.

See also

System procedures sp_who

sp_showpsexe

Description

Displays execution class, current priority, and affinity for all client sessions running on Adaptive Server.

Syntax

sp_showpsexe [*spid*]

Parameters

spid

is the Adaptive Server session ID for which you want a report. The *spid* must belong to the application or login executing sp_showpsexe. Use sp_who to list spids.

Examples

Example 1 Displays execution class, current priority, and affinity for all current client sessions:

```
sp_showpsexe

spid  appl_name   login_name
 exec_class current_priority task_affinity
----- ----------- ----------- ---------- ---------------- -------------
    1 isql        sa          EC1        HIGH             NONE
    5             NULL        NULL       LOW              NULL
    7 ctisql      sa          EC2        MEDIUM           NONE
    8 ctisql      sa          EC2        MEDIUM           NONE
```

Example 2 Displays the application name, login name, current priority, and engine affinity of the process with spid 5:

```
sp_showpsexe 5
```

Usage

- sp_showpsexe displays execution class, current priority, and affinity for all sessions (objects with an *spid*). For more information, see the Performance and Tuning Guide.
- If the *spid* is NULL or absent, sp_showpsexe reports on all sessions currently running on Adaptive Server.
- sp_showpsexe does not report information for the following system processes: deadlock, checkpoint, network, auditing, and mirror handlers. It does display information for the housekeeper *spid*.

Permissions

Any user can execute sp_showpsexe.

See also

System procedures sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showcontrolinfo, sp_showexeclass, sp_unbindexeclass

sp_spaceused

Description

Displays estimates of the number of rows, the number of data pages, the size of indexes, and the space used by a specified table or by all tables in the current database.

Syntax

sp_spaceused [*objname* [,1]]

Parameters

objname
is the name of the table on which to report. If omitted, a summary of space used in the current database appears.

1
prints separate information on the table's indexes and text/image storage.

Examples

Example 1 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:

```
sp_spaceused titles

name       rowtotal   reserved   data    index_size   unused
---------- ---------  ---------  ------- ----------   ----------
titles     18         46 KB      6 KB    4 KB         36 KB
```

Example 2 In addition to information on the titles table, prints information for each index on the table:

```
sp_spaceused titles, 1

index_name           size       reserved   unused
 -------------------- ---------- ---------- ----------
 titleidind           2 KB       32 KB      24 KB
 titleind             2 KB       16 KB      14 KB

 name       rowtotal   reserved   data    index_size unused
---------- --------  ---------  ------- ---------- ----------
titles           18      46 KB     6 KB       4 KB      36 KB
```

Example 3 Displays the space taken up by the text/image page storage separately from the space used by the table. The object name for text/image storage is "t" plus the table name:

```
sp_spaceused blurbs,1

index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
blurbs               0 KB       14 KB      12 KB
tblurbs              14 KB      16 KB      2 KB
```

```
name       rowtotal reserved    data    index_size unused
---------- -------- ----------- ------- ---------- ----------
blurbs            6       30 KB    2 KB      14 KB      14 KB
```

Example 4 Prints a summary of space used in the current database:

```
sp_spaceused
```

```
database_name   database_size
--------------- --------------------------
master          5 MB
reserved    data        index_size  unused
---------   ---------   ----------- --------
2176 KB     1374 KB     72 KB       730 KB
```

Example 5 Reports on the amount of space reserved and the amount of space available for the transaction log:

```
sp_spaceused syslogs
```

```
name       rowtotal   reserved   data    index_size   unused
---------- ---------  ---------  ------- ----------   ----------
syslogs    Not avail. 32 KB     32 KB   0 KB         0 KB
```

Usage

- sp_spaceused displays estimates of the number of data pages, space used by a specified table or by all tables in the current database, and the number of rows in the tables. 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 one of these commands executes. Always use select count(*) if you need exact row counts.
- sp_spaceused reports on the amount of space affected by tables, clustered indexes, and nonclustered indexes.
- The amount of space allocated (reserved) reported by sp_spaceused is a total of the data, index size, and available (unused) space.
- Space used by text and image columns, which are stored as separate database objects, is reported separately in the index_size column and is included in the summary line for a table. The object name for text/image storage in the index_size column is "t" plus the table name.

- When used on syslogs, sp_spaceused reports rowtotal as “Not available”. See Example 5.

Permissions

Any user can execute sp_spaceused.

See also

Catalog stored procedures sp_statistics

Commands create index, create table, drop index, drop table

System procedures sp_helpindex

sp_ssladmin

Description

Adds, deletes, or displays a list of server certificates for Adaptive Server.

Syntax

sp_ssladmin [addcert, *certificate_path* [, *password* | NULL]]

sp_ssladmin [dropcert, *certificate_path*]

sp_ssladmin [lscert]

sp_ssladmin [help]

Parameters

addcert
adds a certificate for the local server in the certificates file.

certificate_path
specifies the absolute path to the certificates file on the local server.

password
the password that is used to encrypt the private key when adding a new server certificate to the certificates file.

NULL
used to require an attended atart-up of Adaptive Server by requesting the password during start-up from the command line.

dropcert
deletes the certficate from the certificate file.

lscert
lists the certificates in the certificate file.

help
displays online help for sp_ssladmin.

Examples

Example 1 This adds an entry for the local server, Server1.crt, in the certificates file in the absolute path to */sybase/ASE-12_5/certificates* (*x:\sybase\ASE-12_5\certificates* on Windows). The private key is encrypted with the password "mypassword". The password should be the one specified when you created the private key:

```
sp_ssladmin addcert, "/sybase/ASE-12_5/certificates/Server1.crt",
    "mypassword"
```

Example 2 Deletes the certificate, Server1.crt from the certificates file located in */sybase/ASE-12_5/certificates* (*x:\sybase\ASE-12_5\certificates* on Windows):

```
sp_ssladmin dropcert , "/sybase/ASE-12_5/certificates/Server1.crt"
```

Example 3 Lists of all server certificates on the local server:

```
sp_ssladmin lscert
go

certificate_path
----------------------------------------
/sybase/ASE-12_5/certificates/Server1.crt
```

Usage

- The Adaptive Server listener must present to the client a certificate. The common name in the certificate must match the common name used by the client in the interfaces file. If they do not match, the server authentication and login fail.
- When NULL is specified as the password, dataserver must be started with a -y flag. This flag prompts the administrator for the private-key password at the command line.
- The use of NULL as the password is intended to protect passwords during the intitial configuration of SSL, before the SSL encrypted session begins.

  After restarting Adaptive Server with an SSL connection established, use sp_ssladmin again, this time using the actual password. The password is then encrypted and stored by Adaptive Server. Any subsequent starts of Adaptive Server from the command line would use the encrypted password; you do not have to specify the password on the command line during start up.
- You can specify "localhost" as the *hostname* in the *interfaces* file (*sql.ini* on Windows) to prevent clients from connecting remotely. Only a local connection can be established, and the password is never transmitted over a network connection.

Permissions

You must have the System Security Officer role to use sp_ssladmin.

sp_syntax

Description

Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines for Adaptive Server, depending on which products and corresponding sp_syntax scripts exist on your server.

Syntax

sp_syntax *word* [, *mod*][, *language*]

Parameters

word
: is the name or partial name of a command or routine; for example, “help”, to list all system procedures providing help. To include spaces or Transact-SQL reserved words, enclose the word in quotes.

mod
: 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 be retrieved. *language* must be a valid language name in the syslanguages table.

Examples

Example 1 Displays all sp_syntax modules available on your server:

```
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]]
```

Example 2 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:

```
sp_syntax "disk"
```

Usage	• The text for sp_syntax is in the database sybsyntax. Load sp_syntax and the sybsyntax database onto Adaptive Server with the installation script described in configuration documentation for your platform. If you cannot access sp_syntax, see your System Administrator for information about installing it on your server. • You can use wildcard characters within the command name you are searching for. However, if you are looking for a command or function that contains the literal "_", you may get unexpected results, since the underscore wildcard character represents any single character.
Permissions	Any user can execute sp_syntax.
Tables used	sybsyntax..sybsyntax
See also	**System procedures** sp_helpdb

sp_sysmon

Description

Displays performance information.

Syntax

```
sp_sysmon begin_sample
sp_sysmon { end_sample | interval } [, section [, applmon] ]
sp_sysmon { end_sample | interval } [, applmon ]
```

Parameters

begin_sample
: starts sampling. You cannot specify a section when you specify begin_sample.

end_sample
: ends sampling and prints the report.

interval
: specifies the time period for the sample. It must be in HH:MM:SS form, for example “00:20:00”.

section
: is the abbreviation for one of the sections printed by sp_sysmon. Table 1-20 lists the values and corresponding names of the report sections.

Table 1-20: sp_sysmon report sections

Report section	Parameter
Application Management	appmgmt
Data Cache Management	dcache
Disk I/O Management	diskio
ESP Management	esp
Index Management	indexmgmt
Kernel Utilization	kernel
Lock Management	locks
Memory Management	memory
Metadata Cache Management	mdcache
Monitor Access to Executing SQL	monaccess
Network I/O Management	netio
Parallel Query Management	parallel
Procedure Cache Management	pcache
Recovery Management	recovery
Task Management	taskmgmt
Transaction Management	xactmgmt
Transaction Profile	xactsum
Worker Process Management	wpm

applmon

specifies whether to print application detail, application and login detail, or no application detail. The default is to omit the application detail. Valid values are listed in Table 1-21.

Table 1-21: Values for applmon parameter to sp_sysmon

Parameter	Information reported
appl_only	CPU, I/O, priority changes and resource limit violations by application name.
appl_and_login	CPU, I/O, priority changes and resource limit violations by application name and login name.
no_appl	Skips the by application or by login section of the report. This is the default.

This parameter is only valid when printing the full report and when you specify appmgmt for the *section*.

Examples

Example 1 Prints monitor information after 10 minutes:

```
sp_sysmon "00:10:00"
```

Example 2 Prints only the "Disk Management" section of the sp_sysmon report after 5 minutes:

```
sp_sysmon "00:05:00", diskio
```

Example 3 Starts the sample, executes procedures and a query, ends the sample, and prints only the "Data Cache" section of the report:

```
sp_sysmon begin_sample
go
execute proc1
go
execute proc2
go
select sum(total_sales) from titles
go
sp_sysmon end_sample, dcache
go
```

Example 4 Prints the full report and includes application and login detail for each login:

```
sp_sysmon "00:05:00", @applmon = appl_and_login
```

Usage

- sp_sysmon displays information about Adaptive Server performance. It sets internal counters to 0, then waits for the specified interval while activity on the server causes the counters to be incremented. When the interval ends, sp_sysmon prints information from the values in the counters. See the *Performance and Tuning Guide* for more information.
- To print only a single section of the report, use the values listed in Table 1-21 for the second parameter.
- If you use sp_sysmon in batch mode, with begin_sample and end_sample, the time interval between executions must be at least one second. You can use waitfor delay "00:00:01" to lengthen the execution time of a batch.
- During the sample interval, results are stored in signed integer values. Especially on systems with many CPUs and high activity, these counters can overflow. If you see negative results in your sp_sysmon output, reduce your sample time.

Permissions

Only a System Administrator can execute sp_sysmon.

sp_tempdb

Description

sp_tempdb allows users to:

- Create the default temporary database group
- Bind temporary databases to the default temporary database group
- Bind users and applications to the default temporary database group or to specific temporary databases

These bindings are stored in the sysattributes table in master database.

sp_tempdb provides the binding interface for maintaining bindings in sysattributes that are related to the multiple temporary database.

Syntax

```
sp_tempdb [
    [ { create | drop } , groupname ] |
    [ { add | remove } , tempdbname, groupname ] |
    [ { bind, objtype, objname, bindtype, bindobj [, scope, hardness ] } |
      { unbind, objtype, objname [, scope ] } ] |
    [ unbindall_db, tempdbname ] |
    [ show [, "all" | "gr" | "db" | "login" | "app" [, name ] ] |
    [ who, dbname ]
    [ help ]
]
```

Parameters

create
creates the default temporary database group.

drop
drops a database group.

groupname
is the default database group. Use “default”.

add
adds temporary databases to the default temporary database group.

remove
removes temporary databases from the default temporary database group.

tempdbname
is the name of the temporary database you are adding or removing.

bind
binds logins and applications to temporary databases or the default temporary database group.

unbind
: unbinds logins and applications to temporary databases or the default temporary database group.

objtype
: is the object type. Valid values are:

- login_name (or LG)
- application_name (or AP)

Values are not case-sensitive.

objname
: is the name of the object you bind or unbind.

bindtype
: is the bind type. Valid values are:

- group (or GR)
- database (or DB)

Values are not case-sensitive.

bindobj
: is the name of the object being bound, and is either a group or a database depending on the *bindtype*.

scope
: NULL.

hardness
: is hard, soft, or NULL. The default is soft.

When you set the value of *hardness* to hard, a failure to assign a temporary database according to the binding results in a failure of the login. When you set the value to soft, such a failure results in the assignment of the system tempdb.

unbindall_db
: removes all login and application bindings for a given temporary database. It does not remove any database to group memberships. The *tempdbname* variable is required with this option.

Existing assignments to active sessions are not affected by this operation.

show

displays information stored in the sysattributes table about the existing groups, group members, login and application bindings, and active sessions that are assigned to a given database. The values are:

- all or no argument – displays the default temporary database group, all database-to-group memberships, and all login and application bindings.
- gr – displays the default temporary database group. sp_tempdb show displays all temporary databases bound to the default temporary database group whether you specify "default" for the *name* option or not.
- db – displays all databases to group memberships. If you provide *name*, then only the database to group memberships for the database *name* are printed.
- login – displays all login bindings where login is not NULL. If you provide *name*, then only the bindings for the login *name* are printed.
- app – displays all bindings where the application is not NULL. If you provide *name*, then the bindings for the application *name* are printed.

Note tempdb is always part of the default database group.

---

who
displays all active sessions assigned to the given temporary database. When using the who parameter, you must use:

- *dbname* – the name of a temporary database. If you provide a nontemporary database name for *dbname*, sp_tempdb who executes, but does not report any active sessions bound to it.

help
displays usage information. Executing sp_tempdb without specifying a command is the same as executing sp_tempdb "help".

Examples

Example 1 Adds mytempdb1 to the default group:

```
sp_tempdb add, mytempdb1, "default"
```

Example 2 Removes mytempdb1 from the default group:

```
sp_tempdb remove, mytempdb1, "default"
```

Example 3 Binds login "sa" to the default group:

```
sp_tempdb bind, lg, sa, GR, "default"
```

The value for objtype in this example is login_name. You can substitute login_name with lg or LG.

The value for bindtype in this example is group. You can substitute group with gr or GR.

Example 4 Changes the previous binding of login "sa" from the default group to mytempdb1:

```
sp_tempdb bind, lg, sa, DB, mytempdb1
```

The value for bindtype in this example is database. You can substitute database with db or DB.

Example 5 Binds isql to mytempdb1:

```
sp_tempdb bind, ap, isql, DB, mytempdb1
```

The value for objtype in this example is application_name. You can substitute application_name with ap or AP.

Example 6 Changes the previous binding of isql from mytempdb1 to the default group:

```
sp_tempdb bind, ap, isql, GR, "default"
```

Example 7 Removes the bindings of login "sa" and application "isql".

```
sp_tempdb unbind, lg, sa
```

```
sp_tempdb unbind, ap, isql
```

Example 8 Removes all login and application bindings for the mytempdb1 database:

```
sp_tempdb unbindall_db, mytempdb1
```

Example 9 Demonstrates the sp_temp show command. A selection of the different variations is chosen, and abbreviated sample output is displayed.

```
sp_tempdb show
```

```
Temporary Database Groups
------------------------------
default

Database                        GroupName
------------------------------ ---------------
tempdb                          default
mytempdb                        default
mytempdb1                       default
mytempdb2                       default
```

```
mytempdb3                       default

Login   Application  Group    Database    Hardness
------- ------------ -------- ----------- --------
NULL   isql          default  NULL        SOFT
sa      NULL          NULL     mytempdb3   HARD
```

Example 10 Displays the default temporary database group:

```
sp_tempdb show, gr
```

```
Temporary Database Groups
-------------------------------
default
```

Example 11 Displays all the temporary database group names that are bound to the default group:

```
sp_tempdb show, gr, "default"
```

```
Member Databases
-------------------------------
tempdb
mytempdb
mytempdb1
mytempdb2
mytempdb3
```

Example 12 Displays all the databases-to-group memberships:

```
sp_tempdb show, db
```

```
Database             Group
-------------------- ----------------
tempdb               default
mytempdb             default
mytempdb1            default
mytempdb2            default
mytempdb3            default
```

Example 13 Displays all the databases-to-group memberships for the mytempdb1 database.

```
sp_tempdb show, db, mytempdb1
```

```
Database             Group
-------------------- ----------------
```

```
mytempdb1              default
```

Example 14 Displays all the login bindings where login is not NULL:

```
sp_tempdb show, login

Login   Application   Group   Database    Hardness
------- ------------- ------- ----------- --------

sa      NULL          NULL    mytempdb3   HARD
```

Example 15 Displays all active sessions that are assigned to the system tempdb:

```
sp_tempdb who, tempdb

spid   loginame
------ ------------------------------
2      NULL
3      NULL
4      NULL
5      NULL
6      NULL
7      NULL
8      NULL
```

Example 16 Displays all active sessions that are assigned to the mytempdb3 user-created temporary database:

```
sp_tempdb who, mytempdb3

spid   loginame
------ ------------------------------
17      sa
```

Example 17 Displays usage information:

```
sp_tempdb help

Usage:
sp_tempdb 'help'
sp_tempdb 'create', <groupname>
sp_tempdb 'drop', <groupname>
sp_tempdb 'add', <tempdbname>, <groupname>
sp_tempdb 'remove', <tempdbname>, <groupname>
sp_tempdb 'bind', <objtype>, <objname>, <bindtype>, <bindobj>, <scope>,
          <hardness>
```

```
sp_tempdb 'unbind', <objtype>, <objname>, <scope>
sp_tempdb 'unbindall_db', <tempdbname>
sp_tempdb 'show', <command>, <name>
sp_tempdb 'who', <dbname>

<objtype> = ['LG' ('login_name') | 'AP' ('application_name')];
<bindtype> =['GR' ('group') | 'DB' ('database')]
<hardness> = ['hard' | 'soft']
<command> = ['all' | 'gr' | 'db' | 'login' | 'app']
```

Usage

create and drop

When using the sp_tempdb create stored procedure, the *groupname* variable:

- Must be a valid identifier
- Cannot already exist

The default group is the system-generated group, of which tempdb is always a member. This default group is present if you:

- Upgrade using the Adaptive Server containing this feature, or
- Create a new master device.

If the default group is not present, you can create it by using:

```
sp_tempdb create, "default"
```

An error message displays if you attempt to create a default group that already exists.

add and remove

To add a temporary database to the default temporary database group, both the temporary database and the group name must already exist. When you use sp_tempdb add to add a *tempdbname* to a set of databases that are members of the default temporary database group, *tempdbname* becomes available for round-robin assignment from within that group.

Note sp_tempdb add fails if *tempdbname* is not already part of the global list of available temporary databases in Adaptive Server.

User-created temporary databases need not belong to the default temporary database group. The system tempdb is implicitly a member of the default group.

If you try to add a temporary database to the default temporary database group when it is already a part of that group, you get an error message, and no changes take place in sysattributes.

Permissions

By default, only the System Administrator or users with the SA role can execute sp_tempdb.

sp_thresholdaction

Description

Executes automatically when the number of free pages on the log segment falls below the last-chance threshold, unless the threshold is associated with a different procedure. Sybase does not provide this procedure.

Syntax

When a threshold is crossed, Adaptive 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 logical pages.

@status
is 1 for the last-chance threshold; 0 for all other thresholds.

Examples

Creates a threshold procedure for the last-chance threshold that dumps the transaction log to a tape device:

```
create procedure sp_thresholdaction
        @dbname varchar(30),
        @segmentname varchar(30),
        @space_left int,
        @status int
as
        dump transaction @dbname to tapedump1
```

Usage

- sp_thresholdaction must be created by the Database Owner (in a user database), or a System Administrator (in the sybsystemprocs database), or 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, Adaptive Server searches for the sp_thresholdaction procedure in the database where the threshold event occurs. If it does not exist in that database, Adaptive Server searches for it in sybsystemprocs. If it does not exist in sybsystemprocs, it searches master. If Adaptive 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 is sent to the error log. To abort these commands rather than suspend 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 processes automatically awaken and complete.
- If fast bcp, writetext, or select into have 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 use dump transaction to truncate the transaction log.
- 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, then use the kill command to kill them.

See also

Commands create procedure, dump transaction

System procedures sp_addthreshold, sp_dboption, sp_dropthreshold, sp_helpsegment, sp_helpthreshold, sp_modifythreshold, sp_who

sp_transactions

Description

Reports information about active transactions.

Syntax

sp_tranactions ["xid", *xid_value*] |
    ["state", {"heuristic_commit" | "heuristic_abort"
    | "prepared" | "indoubt"} [, "xactname"]] |
    ["gtrid", *gtrid_value*]

Parameters

xid_value
is a transaction name from the xactname column of master.dbo.systransactions.

gtrid_value
is the global transaction ID name for a transaction coordinated by Adaptive Server.

Examples

Example 1 Displays general information about all active transactions:

```
sp_transactions

xactkey                        type        coordinator starttime
state             connection dbid   spid   loid
failover                   srvname                        namelen
xactname
------------------------------ ----------- -----------
--------------------
----------------- ---------- ------ ------ -----------
------------------------- ------------------------------ -------
-----------------------------------------------------------------
0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
Begun             Attached        1    1      2
Resident Tx                NULL                               17
$user_transaction
0x00000b1700040000dd6821390001 Remote      ASTC        Jun 1 1999 3:47PM
Begun             NA              0    8      0
Resident Tx                caserv2                         108

00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-aa0
1f04ebb9a-caserv1-caserv1-0002
```

Example 2 Displays detailed information for the specified transaction:

```
sp_transactions "xid",
"00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-aa
01f04ebb9a-caserv1-caserv1-0002"

xactkey                        type        coordinator starttime
state             connection dbid   spid   loid
failover                   srvname                        namelen
```

```
xactname
commit_node            parent_node
gtrid
------------------------------ ---------- -----------
--------------------
---------------- ---------- ------ ------ -----------
-------------------------- ------------------------------ -------
-----------------------------------------------------------------
-------------
-------------
-------------
0x00000b2500080000dd6821960001 External    ASTC        Jun 1 1999 3:47PM
Begun            Attached       1      8         139
Resident Tx              NULL                             108

00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-aa0
1f04ebb9a-caserv1-caserv1-0002

caserv1            caserv1
00000b1700040000dd6821390001-aa01f04ebb9a
```

Example 3 Displays general information about transactions that are in the "prepared" state:

```
sp_transactions "state", "prepared"
```

Example 4 Displays only the transaction names of transactions that are in the "prepared" state:

```
sp_transactions "state", "prepared", "xactname"
```

Example 5 Displays status information for transactions having the specified global transaction ID:

```
sp_transactions "gtrid", "00000b1700040000dd6821390001-aa01f04ebb9a"
```

```
xactkey                        type        coordinator starttime
state            connection dbid   spid   loid
failover                 srvname                         namelen
xactname
commit_node
parent_node
------------------------------ ---------- -----------
--------------------
---------------- ---------- ------ ------ -----------
-------------------------- ------------------------------ -------
-----------------------------------------------------------------
-------------
-------------
```

```
0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
Begun            Attached       1      1           2
Resident Tx               NULL                             17
$user_transaction

caserv1
caserv1
```

Usage

- sp_transactions translates data from the systransactions table to display information about active transactions. systransactions itself comprises data in the syscoordinations table, as well as in-memory information about active transactions.
- sp_transactions with no keywords displays information about all active transactions.
- sp_transactions with the xid keyword displays the gtrid, commit_node, and parent_node columns only for the specified transaction.
- sp_transactions with the state keyword displays information only for the active transactions in the specified state.

  sp_transactions with both xid and xactname displays only the transaction names for transactions in the specified state.
- sp_transactions with the gtrid keyword displays information only for the transactions with the specified global transaction ID.
- sp_transactions replaces the sp_xa_scan_xact procedure provided with XA-Library and XA-Server products.
- See *Using Adaptive Server Distributed Transaction Management Features* for more information.

Column descriptions for *sp_transactions* output

- The xactkey column shows the internal transaction key that Adaptive Server uses to uniquely identify the transaction.
- The type column indicates the type of transaction:
  - “Local” means that the transaction was explicitly started on the local Adaptive Server with a begin transaction statement.
  - “Remote” indicates a transaction executing on a remote Adaptive Server.

  - "External" means that the transaction has an external coordinator associated with it. For example, transactions coordinated by a remote Adaptive Server, MSDTC, or an X/Open XA transaction manager are flagged as "External."
  - "Dtx_State" is a special state for distributed transactions coordinated by Adaptive Server. It indicates that a transaction on the local server was either committed or aborted, but Adaptive Server has been unable to resolve a branch of that transaction on a remote participant. This may happen in cases where Adaptive Server loses contact with a server it is coordinating.
- The coordinator column indicates the method or protocol used to manage a distributed transaction:

sp_transactions "coordinator" value	**Meaning**
None	Transaction is not a distributed transaction and does not require a coordinating protocol.
ASTC	Transaction is coordinated using the Adaptive Server transaction coordination services.
XA	Transaction is coordinated by the X/Open XA-compliant transaction manager via the Adaptive Server XA-Library interface. Such transaction managers include Encina, CICS, and Tuxedo.
DTC	Transaction is coordinated by MSDTC.
SYB2PC	Transaction is coordinated using Sybase two-phase commit protocol.

- The starttime column indicates the time that the transaction started.
- The state column indicates the state of the transaction at the time sp_transactions ran:

sp_transactions "state" value	**Meaning**
Begun	Transaction has begun but no updates have been performed.
Done Command	Transaction completed an update command.
Done	X/Open XA transaction has finished modifying data.
Prepared	Transaction has successfully prepared.
In Command	Transaction is currently modifying data.
In Abort Cmd	Execution of the current command in the transaction has been aborted.
Committed	Transaction has successfully committed, and the commit log record has been written.
In Post Commit	Transaction has successfully committed, but is currently deallocating transaction resources.
In Abort Tran	Transaction is being aborted. This may happen either as a result of an explicit command, or because of a system failure.
In Abort Savept	Transaction is being rolled back to a savepoint.

sp_transactions "state" value	**Meaning**
Begun-Detached	Transaction has begun, but there is no thread currently attached to it.
Done Cmd-Detached	Transaction has finished modifying data, and no thread is currently attached to it.
Done-Detached	Transaction will modify no more data, and no thread is currently attached to it.
Prepared-Detached	Transaction has successfully prepared, and no thread is currently attached to it.
Heur Committed	Transaction has been heuristically committed using the dbcc complete_xact command.
Heur Rolledback	Transaction has been heuristically rolled back using the dbcc complete_xact command.

- The connection column indicates whether or not the transaction is currently associated with a thread:
  - "Attached" indicates that the transaction has an associated thread of control.
  - "Detached" indicates that there is no thread currently associated with the transaction. Some external transaction managers, such as CICS and TUXEDO, use the X/Open XA "suspend" and "join" semantics to associate different threads with the same transaction.
- The dbid column indicates the database ID of the database in which transaction started.
- The spid column indicates the server process ID associated with the transaction. If the transaction is "Detached," the "spid" value is 0.
- The loid column indicates the unique lock owner ID from master.dbo.systransactions.
- The failover column indicates the failover state for the transaction:
  - "Resident Tx" indicates that the transaction started and is executing on the same server. "Resident Tx" is displayed under normal operating conditions, and on systems that do not utilize Adaptive Server high availability features.
  - "Failed-over Tx" is displayed after there has been a failover to a secondary companion server. "Failed-over Tx" means that a transaction originally started on a primary server and reached the prepared state, but was automatically migrated to the secondary companion server (for example, as a result of a system failure on the primary server). The migration of a prepared transaction occurs transparently to an external coordinating service.

    - "Tx by Failover-Conn" indicates that there was an attempt to start the transaction on a designated server, but the transaction was instead started on the secondary companion server. This occurs when the original server has experienced a failover condition.

- The srvname column indicates the name of the remote server on which the transaction is executing. This column is only meaningful for remote transactions. For local and external transactions, srvname is null.
- The namelen column indicates the total length of the *xactname* value.
- *xactname* is the transaction name. For local transactions, the transaction name may be defined as part of the begin transaction command. External transaction managers supply unique transaction names in a variety of formats. For example, X/Open XA-compliant transaction managers supply a transaction ID (*xid*) consisting of a global transaction identifier and a branch qualifier, both of which are stored in *xactname*.
- For transactions coordinated by Adaptive Server, the gtrid column displays the global transaction ID. Transaction branches that are part of the same distributed transaction share the same gtrid. You can use a specific gtrid with the sp_transactions gtrid keyword to determine the state of other transaction branches in the same distributed transaction.

  sp_transactions cannot display the gtrid for transactions that have an external coordinator. For transactions coordinated by an X/Open XA-compliant transaction manager, MSDTC, or SYB2PC, the gtrid column shows the full transaction name supplied by the external coordinator.
- For transactions coordinated by Adaptive Server, the commit_node column indicates the server that executes the outermost block of the distributed transaction. This outermost block ultimately determines the commit status of all subordinate transactions.

  For transactions not coordinated by Adaptive Server, commit_node displays one of the values described in Table 1-22.

Table 1-22: Values for commit_node and parent_node

Value	Meaning
server_name	Commit or parent node is an Adaptive Server with the specified *server_name*.
XATM	Commit or parent node is an X/Open XA-compliant transaction manager.
MSDTCTM	Commit or parent node is MSDTC.
SYB2PCTM	Transaction is coordinated using SYB2PC protocol.

- For transactions coordinated by Adaptive Server, the parent_node column indicates the server that is coordinating the external transaction on the local server.

  For transactions not coordinated by Adaptive Server, parent_node displays one of the values described in Table 1-22.

  Note The values for commit_node and parent_node can be different, depending on the levels of hierarchy in the distributed transaction.

Permissions

Any user can execute sp_transactions.

See also

System procedures sp_lock, sp_who

sp_unbindcache

Description

Unbinds a database, table, index, text object, or image object from a data cache.

Syntax

sp_unbindcache *dbname* [,[*owner*.]*tablename*
    [, *indexname* | "text only"]]

Parameters

dbname
is the name of database to be unbound or the name of the database containing the objects to be unbound.

owner
is the name of the table's owner. If the table is owned by the Database Owner, the owner name is optional.

tablename
is the name of the table to be unbound from a cache or the name of a table whose index, text object, or image object is to be unbound from a cache.

indexname
is the name of an index to be unbound from a cache.

text only
unbinds text or image objects from a cache.

Examples

Example 1 Unbinds the titles table from the cache to which it is bound:

```
sp_unbindcache pubs2, titles
```

Example 2 Unbinds the titleidind index from the from the cache to which it is bound:

```
sp_unbindcache pubs2, titles, titleidind
```

Example 3 Unbinds the text or image object for the au_pix table from the cache to which it is bound:

```
sp_unbindcache pubs2, au_pix, "text only"
```

Example 4 Unbinds the transaction log, syslogs, from its cache:

```
sp_unbindcache pubs2, syslogs
```

Usage

- When you unbind a database or database object from a cache, all subsequent I/O for the cache is performed in the default data cache. All dirty pages in the cache being unbound are written to disk, and all clean pages are cleared from the cache. See the *Performance and Tuning Guide* for more information.
- Cache unbindings take effect immediately and do not require a restart of the server, except with the system tempdb.

- Although you can still use sp_unbindcache on a system tempdb, the binding of the system tempdb is now non-dynamic. Until you restart the server:
    - The changes do not take effect
    - sp_helpcache reports a status of "P" for pending, unless you have explicitly bound the system tempdb to the default data cache, in which case the status as "V" for valid, because by default the system tempdb is already bound to the default datacache.
- When you drop a database, table, or index, its cache bindings are automatically dropped.
- To unbind a database, you must be using the master database. For tables, indexes, text objects, or image objects, you must be using the database where the objects are stored.
- To unbind any system tables in a database, you must be using the database, and the database must be in single-user mode. Use the command:

  ```
  sp_dboption db_name, "single user", true
  ```

  See sp_dboption for more information.
- The following procedures provide information about the bindings for their respective objects: sp_helpdb for databases, sp_help for tables, and sp_helpindex for indexes.
- sp_helpcache prints the names of objects bound to caches.
- sp_unbindcache needs to acquire an exclusive table lock when you are unbinding a table or its indexes to a cache. No pages can be read while the unbinding takes place. If a user holds locks on a table, and you issue sp_unbindcache on that object, the sp_unbindcache task sleeps until the locks are released.
- When you change the cache binding for an object with sp_bindcache or sp_unbindcache, the stored procedures that reference the object are recompiled the next time they are executed. When you change the binding for a database, the stored procedures that reference objects in the database are recompiled the next time they are executed.
- To unbind all objects from a cache, use the system procedure sp_unbindcache_all.

Permissions

Only a System Administrator can execute sp_unbindcache.

See also

System procedures sp_bindcache, sp_dboption, sp_help, sp_helpdb, sp_helpcache, sp_helpdb, sp_helpindex, sp_unbindcache_all

sp_unbindcache_all

Description

Unbinds all objects that are bound to a cache.

Syntax

sp_unbindcache_all *cache_name*

Parameters

cache_name
is the name of the data cache from which objects are to be unbound.

Examples

Unbinds all databases, tables, indexes, text objects and image objects that are bound to pub_cache:

```
sp_unbindcache_all pub_cache
```

Usage

- When you unbind entities from a cache, all subsequent I/O for the cache is performed in the default cache.
- To unbind individual objects from a cache, use the system procedure sp_unbindcache.
- You cannot use sp_unbindcache_all if the system tempdb is bound to pub_cache. If you do, you get an error message, and sp_unbindcache_all rejects the unbind for all objects.

  Use sp_unbindcache to unbind the system tempdb first.
- See sp_unbindcache for more information about unbinding caches.

Permissions

Only a System Administrator can execute sp_unbindcache_all.

See also

System procedures sp_bindcache, sp_helpcache, sp_unbindcache

sp_unbindefault

Description

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 assumed 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 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. It is ignored when unbinding a default from a column.

Examples

Example 1 Unbinds the default from the startdate column of the employees table:

```
sp_unbindefault "employees.startdate"
```

Example 2 Unbinds the default from the user-defined datatype named ssn and all columns of that type:

```
sp_unbindefault ssn
```

Example 3 Unbinds defaults from the user-defined datatype ssn, but does not affect existing columns of that type:

```
sp_unbindefault ssn, futureonly
```

Usage

- 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 the default has 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.

Permissions

Only the object owner can execute sp_unbindefault.

See also

Commands create default, drop default

System procedures sp_bindefault, sp_helptext

sp_unbindexeclass

Description

Removes the execution class attribute previously associated with an client application, login, or stored procedure for the specified scope.

Syntax

sp_unbindexeclass *object_name*, *object_type*, *scope*

Parameters

object_name
is the name of the application, login, or stored procedure for which to remove the association to the execution class.

object_type
identifies the type of *object_name* as ap, lg, or pr for application, login, or stored procedure.

scope
is the application name or the login name for which the unbinding applies for an application or login. It is the stored procedure owner name (user name) for stored procedures.

Examples

Removes the association between "sa" login scoped to application isql and an execution class. "sa" automatically binds itself to another execution class, depending on other binding specifications, precedence, and scoping rules. If no other binding is applicable, the object binds to the default execution class, EC2:

```
sp_unbindexeclass 'sa', 'lg', 'isql'
```

Usage

- The parameters must match an existing entry in the sysattributes system table.
- If you specify a null value for scope, Adaptive Server unbinds the object for which the scope is null, if there is one.
- A null value for scope does not indicate that unbinding should apply to all bound objects.
- When unbinding a stored procedure from an execution class, you must use the name of the stored procedure owner (user name) for the scope parameter.
- Stored procedures can be dropped before or after unbinding.
- A user cannot be dropped from a database if the user owns a stored procedure that is bound to an execution class in that database.
- Unbind objects of type PR before dropping them from the database.
- Unbinding will fail if the associated engine group has no online engines and active processes are bound to the associated execution class.

- Due to precedence and scoping rules, the execution class being unbound may or may not have been in effect for the object called *object_name*. The object automatically binds itself to another execution class, depending on other binding specifications and precedence and scoping rules. If no other binding is applicable, the object binds to the default execution class, EC2.

Permissions

Only a System Administrator can execute sp_unbindexeclass.

See also

System procedures sp_addexeclass, sp_bindexeclass, sp_dropexeclass, sp_showexeclass

Utility isql

sp_unbindmsg

Description	Unbinds a user-defined message from a constraint.
Syntax	sp_unbindmsg *constrname*
Parameters	*constrname* is the name of the constraint from which a message is to be unbound.
Examples	Unbinds a user-defined message from the constraint positive_balance: `sp_unbindmsg positive_balance`
Usage	• 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.
Permissions	Only the object owner can execute sp_unbindmsg.
See also	**System procedures** sp_addmessage, sp_bindmsg, sp_getmessage

sp_unbindrule

Description

Unbinds a rule from a column or from a user-defined datatype.

Syntax

sp_unbindrule *objname* [, futureonly [, “accessrule” | “all”]]

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 assumed to be a user-defined datatype. Unbinding a rule from a user-defined datatype also unbinds it from columns of the same type. Columns that are already bound to a different rule are unaffected.

futureonly

prevents columns of the specified user-defined datatype from losing their rules. It is ignored when unbinding a rule from a column.

accessrule

indicates that you are unbinding the access rule bound to *objname*.

all

specifies that you are unbinding all rules bound to *objname*.

Examples

Example 1 Unbinds the rule from the startdate column of the employees table:

```
sp_unbindrule "employees.startdate"
```

Example 2 Unbinds the rule from the user-defined datatype named def_ssn and all columns of that type:

```
sp_unbindrule def_ssn
```

Example 3 The user-defined datatype ssn no longer has a rule, but existing ssn columns are unaffected:

```
sp_unbindrule ssn, futureonly
```

Example 4 You can use the all parameter to unbind both accesss rules and domain rules. For example, to unbind all the access rules and domain rules on the publishers table:

```
sp_unbindrule publishers, null, "all"
```

To unbind the access rule from a user-defined datatype for subsequent uses of this datatype, issue:

```
sp_unbindrule def_ssn, futureonly, "accessrule"
```

To unbind both access rules and domain rules for subsequent uses of this datatype, issue:

```
sp_unbindrule def_ssn, futureonly, "all"
```

Example 5 This access rule is bound to the publishers table:

```
sp_bindrule empl_access, "publishers.pub_id"
```

To unbind this rule, issue the following:

```
sp_unbindrule "empl_access", NULL, "accessrule"
```

Usage

- Executing sp_unbindrule removes a rule from a column or from a user-defined datatype in the current database. If you do not want to unbind the rule from existing objname columns, use 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 form "*table.column*".
- The rule is unbound from all existing columns of the user-defined datatype unless the rule has been changed or the value of the optional second parameter is futureonly.
- To display the text of a rule, execute sp_helptext with the rule name as the parameter.

Permissions

Only the object owner can execute sp_unbindrule.

See also

Commands create rule, drop rule

System procedures sp_bindrule, sp_helptext

sp_volchanged

Description

Notifies the Backup Server that the operator performed the requested volume handling during a dump or load.

Syntax

```
sp_volchanged session_id, devname, action
    [, fname [, vname]]
```

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.

devname

is the device on which a new volume was mounted. Use the *@devname* parameter specified in the Backup Server's volume change request. If the Backup Server is not located on the same machine as the Adaptive 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.

fname

is the file to be loaded. 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.

vname

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 *vname* 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 *vname* to confirm that the correct tape has been mounted. If you do not specify a *vname* 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

The operator changes the tape, then issues the command:

```
sp_volchanged 8, "/dev/nrmt4", RETRY
```

The following 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' }
```

Usage

- 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 the 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 Adaptive Server that can communicate with the Backup Server performing the dump or load. The operator does not have to log into the Adaptive 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 and executes sp_volchanged. Table 1-23 illustrates this process.

Table 1-23: Changing tape volumes on a UNIX system

Sequence	Operator, using isql	Adaptive Server	Backup Server
1	• Issues the dump database command		

Sequence	Operator, using isql	Adaptive Server	Backup Server
2		• Sends dump request to Backup Server	
3			• Receives dump request message from Adaptive Server • Sends message for tape mounting to operator • Waits for operator's reply
4	• Receives volume change request from Backup Server • Mounts tapes • Executes sp_volchanged		
5			• Checks tapes • If tapes are okay, begins dump • When tape is full, sends volume change request to operator
6	• Receives volume change request from Backup Server • Mounts tapes • Executes sp_volchanged		
7			• Continues dump • When dump is complete, sends messages to operator and Adaptive Server
8	• Receives message that dump is complete • Removes and labels tapes	• Receives message that dump is complete • Releases locks • Completes the dump database command	

Permissions

Any user can execute sp_volchanged.

See also

Commands dump database, dump transaction, load database, load transaction

Utility isql

sp_who

Description

Reports information about all current Adaptive Server users and processes or about a particular user or process.

Syntax

sp_who [*loginame* | "*spid*"]

Parameters

loginame
is the Adaptive Server login name of the user you are requesting a report on.

spid
is the number of the process you are requesting a report on. Enclose process numbers in quotes (Adaptive Server expects a char type).

Examples

Example 1 Reports on the processes running on Adaptive Server. Process 11 (a select on a table) is blocked by process 8 (a begin transaction followed by an insert on the same table). For process 8, the current *loginame* is "robert", but the original *loginame* is "sa". Login "sa" executed a set proxy command to impersonate the user "robert":

```
sp_who

fid spid  status    loginame origname hostname blk_spid  dbname
       cmd                blk_xloid
--- ----- --------   -------- -------- -------- -------   ------
       ------------------ ---------
  0    1  recv sleep  bird     bird     jazzy    0       master
       AWAITING COMMAND   0x0000ed92
  0    2  sleeping    NULL     NULL              0       master
       NETWORK HANDLER    0x0000ed92
  0    3  sleeping    NULL     NULL              0       master
       MIRROR HANDLER     0x0000ed92
  0    4  sleeping    NULL     NULL              0       master
       AUDIT PROCESS      0x0000ed92
  0    5  sleeping    NULL     NULL              0       master
       CHECKPOINT SLEEP   0x0000ed92
  0    6  recv sleep  rose     rose     petal    0       master
       AWAITING COMMAND   0x0000ed92
  0    7  sleeping    NULL     NULL     actor    0    sybsystemdb
       ASTC HANDLER       0x0000ed92
  0    8  running     robert   sa       helos    0       master
       SELECT             0x0000ed92
  0    9  send sleep  daisy    daisy    chain    0       pubs2
       SELECT             0x0000ed92
  0   10  alarm sleep lily     lily     pond     0       master
       WAITFOR            0x0000ed92
  0   11  lock sleep  viola    viola    cello    8       pubs2
       SELECT            0x0000ed92
```

Example 2 Reports on the processes being run by the user “victoria”:

```
sp_who victoria
```

Example 3 Reports what Adaptive Server process number 17 is doing:

```
sp_who "17"
```

Example 4 Reports on the processes running on Adaptive Server. Although no user processes other than sp_who are running, the server still shows activity. During idle cycles, the housekeeper wash task moves dirty buffers into the buffer wash region, the housekeeper chores task performs other maintenance tasks. The housekeeper garbage collection task , which cleans up data that was logically deleted and resets the rows so that tables have space again, operates at the priority level of the ordinary user.

```
sp_who

fid spid  status    loginame origname hostname blk_spid dbname
       cmd              block_xloid
---- ---- --------  -------- -------- -------- -------- ------
       ---------------- -----------
  0    1  running   sa       sa       helos    0        master
       SELECT             0
  0    2  sleeping  NULL     NULL              0        master
       NETWORK HANDLER    0
  0    3  sleeping  NULL     NULL              0        master
       DEADLOCK TUNE      0
  0    4  sleeping  NULL     NULL              0        master
       MIRROR HANDLER     0
  0    5  sleeping  NULL     NULL     actor    0        master
       ASTC HANDLER       0
  0    6  sleeping  NULL     NULL              0        master
       CHECKPOINT SLEEP   0
  0    5  sleeping  NULL     NULL              0        master
       HK WASH            0
  0    5  sleeping  NULL     NULL              0        master
       HK GC              0
  0    5  sleeping  NULL     NULL              0        master
       HK CHORES           0
  0    5  sleeping  NULL     NULL              0        master
       HK WASH            0
```

Example 5 Reports on a system-induced rollback, either of a transaction or a command.

```
sp_who
fid spid  status    loginame origname hostname blk_spid dbname
       cmd              block_xloid
  0   11  runnable   sa       sa     copperhead  0         db
     ROLLBACK                 0
```

Usage

- sp_who reports information about a specified user or Adaptive Server process.
- Without parameters, sp_who reports which users are running what processes in all databases.
- The columns returned by sp_who are:
  - fid – identifies the family (including the coordinating process and its worker processes) to which a lock belongs. For more information, see sp_familylock.
  - spid – identifies the process number. A System Administrator can use this number with the Transact-SQL kill command to stop the process.
  - status – indicates whether the process is running or sleeping.
  - loginame – the login or alias of the user who started the process. For all system processes, loginame is NULL.
  - origname – If the loginame is an alias, origname shows the real login name. If not, origname shows the same information as loginame.
  - hostname – the name of the server on which the database resides.
  - blk_spid – contains the process IDs 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 needed by another process.
  - dbname – indicates the name of the database on which the process is running.
  - cmd – identifies the command or process currently being executed. Evaluation of a conditional statement, such as an if or while loop, returns `cond`.
  - block_xloid – identifies the unique lock owner ID of a blocking transaction.
- Running sp_who on a single-engine server shows the sp_who process currently running and all other processes that are runnable or in one of the sleep states. In multi-engine servers, there can be a “running” process for each engine.

- If you enable mirrored disks or remote procedure calls, the mirror handler and the site handler also appear in the report from sp_who.

Permissions

Any user can execute sp_who.

See also

Commands kill

System procedures sp_familylock, sp_lock

CHAPTER 2 Catalog Stored Procedures

This chapter describes catalog stored procedures, which retrieve information from the system tables in tabular form.

Topics covered are:

Topics	Page
Overview	473
Specifying optional parameters	474
Pattern matching	475
System procedure tables	475
ODBC datatypes	476

Overview

Table 2-1 lists the catalog stored procedures that are covered in this chapter.

Table 2-1: Catalog stored procedures

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 the databases in Adaptive Server.
sp_datatype_info	Returns information about a particular datatype or about all supported datatypes.
sp_fkeys	Returns information about foreign key constraints created in the current database with the create table or alter table command.
sp_pkeys	Returns information about primary key constraints created for a single table with the create table or alter table command.
sp_server_info	Returns a list of Adaptive Server attribute names and current values.
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.

Procedure	Description
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.

Catalog stored procedures retrieve information from the system tables in tabular form.

The catalog stored procedures, created by installmaster at installation, are located in the sybsystemprocs database and are owned by the System Administrator.

Many of them can be run from any database. If a catalog stored procedure is executed from a database other than sybsystemprocs, it retrieves information from the system tables in the database from which it was executed.

All catalog stored procedures execute at isolation level 1.

All catalog stored procedures report a return status. For example, this means that the procedure executed successfully. The examples in this book do not include the return status:

```
return status = 0
```

Specifying optional parameters

If a parameter value for a catalog stored 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 catalog stored procedure parameters. Doing so may produce unexpected results.

In many cases, it is more convenient to supply parameters to the catalog stored procedures in the form:

```
@parametername = value
```

than to supply all 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"
```

If you specify more parameters then the number of parameters expected by the system procedure, Adaptive Server ignores the extra parameters.

Pattern matching

Adaptive Server offers a wide range of pattern matching through regular expressions. However, for maximum interoperability, assume only SQL standards pattern matching (the % and _ wildcard characters).

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 in the sybsystemprocs database to convert internal system values such as status bits into human-readable format.

The catalog stored procedures sp_column_privileges and sp_table_privileges create and then drop temporary tables.

ODBC datatypes

Table 2-2 and Table 2-3 list the datatype code numbers and matching datatype names returned by sp_columns and sp_sproc_columnsin the “data_type” column. The source for the description is the Open Database Connectivity (ODBC) Application Programming Interface (API).

Table 2-2: Code numbers for ODBC datatypes

Datatype	Code #
char	1
decimal	3
double precision	8
float	6
integer	4
numeric	2
real	7
smallint	5
varchar	12

Table 2-3: Code numbers for extended datatypes

Datatype	Code #
bigint	-5
binary (bit datatype)	-2
bit	-7
date	9
java.lang.Object	1111
long varbinary	-4
long varchar	-1
time	10
timestamp	11
tinyint	-6
varbinary (bit-varying datatype)	-3

sp_column_privileges

Description

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. The use of wildcard characters in pattern matching is not supported.

table_owner
is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table's owner, sp_column_privileges looks 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. 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 wildcard characters to request information for more than one column. If you do not specify a column name, permissions information for all columns in the specified table is returned.

Examples

```
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       dbo           SELECT       YES
pubs2               dbo            discounts          discounttype
     dbo       dbo           UPDATE       YES
pubs2               dbo            discounts          discounttype
     dbo       dbo            REFERENCE   YES
pubs2               dbo            discounts          discounttype
     dbo       guest       SELECT        NO
pubs2               dbo            discounts          discounttype
     dbo       guest       UPDATE        NO
pubs2               dbo            discounts          discounttype
     dbo       guest       REFERENCE     NO
```

Usage

- The results set for sp_column_privileges is:

Column	Datatype	Description
table_qualifier	varchar(32)	The name of the database in which the table specified for the *table_name* parameter is stored.
table_owner	varchar(32)	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
table_name	varchar(32)	The name specified for the *table_name* parameter. This value cannot be NULL.
column_name	varchar(32)	The specified column name. If no column name was specified in the statement, the results include all columns in the specified table.
grantor	varchar(32)	The name of the database user who has granted permissions on column_name to grantee. This value cannot be NULL.
grantee	varchar(32)	The name of the database user who was granted permissions on column_name by grantor. This value cannot be NULL.
privilege	varchar(32)	Identifies the column privilege. May be one of the following: • SELECT – The grantee is permitted to retrieve data for the column. • 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, and NULL.

Permissions

Any user can execute sp_column_privileges.

sp_columns

Description

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 wildcard characters to request information about more than one table.

table_owner
is the owner of the table or view. Use wildcard characters to request information about tables owned by more than one user. If you do not specify a table owner, sp_columns looks 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 wildcard characters to request information about more than one column.

Examples

Example 1 Displays information about all columns in the publishers table that begin with “p”:

```
sp_columns "publishers", null, null, "p%"

table_qualifier  table_owner  table_name  column_name data_type type_name
  precision  length  scale  radix  nullable  remarks ss_data_type colid
---------------- -----------  ----------- ----------- --------- ----------
  ---------   ------  -----  -----   --------  ------- ------------ ----
pubs2            dbo         publishers   pub_id       1       char
  NULL        4       NULL   NULL    0        NULL    47            1
pubs2            dbo         publishers   pub_name     12      varchar
  NULL      40        NULL   NULL    1        NULL    39            2
```

Example 2 Displays information about all columns beginning with “st” in tables that begin with “s”:

```
sp_columns "s%", null, null, "st%"
```

Usage

- The results set for sp_columns is:

Column	Datatype	Description
table_qualifier	varchar(32)	The name of the database in which the table specified for the *table_name* parameter is stored.
table_owner	varchar(32)	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
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 datatypes.
nullable	smallint	The value 1 means NULL is possible; 0 means NOT NULL.
remarks	varchar(254)	
ss_data_type	smallint	An Adaptive Server datatype.
colid	tinyint	A column appended to the results set.
column_def	varchar(255)	
sql_data_type	smallint	
sql_datetime_sub	smallint	
char_octet_length	int	
ordinal_position	int	
is_nullable	varchar(3)	

- sp_columns reports the type_name as float, and data_type as 6 for columns defined as double precision. The Adaptive Server double precision datatype is a float implementation supports the range of values as specified in the ODBC specifications.

Permissions

Any user can execute sp_columns.

sp_databases

Description

Returns a list of databases in Adaptive Server.

Syntax

sp_databases

Parameters

None.

Examples

```
sp_databases

database_name        database_size    remarks
----------------     -------------    ------------
master                  5120          NULL
model                   2048          NULL
mydb                    2048          NULL
pubs2                   2048          NULL
sybsecurity             5120          NULL
sybsystemprocs         16384          NULL
tempdb                  2048          NULL
```

Usage

- The results set for sp_databases is:

Column	Datatype	Description
database_name	char(32)	NOT NULL database name.
database_size	int	Size of database, in kilobytes.
remarks	varchar(254)	Adaptive Server always returns NULL.

Permissions

Any user can execute sp_databases.

sp_datatype_info

Description
Returns information about a particular ODBC datatype or about all ODBC datatypes.

Syntax
sp_datatype_info [*data_type*]

Parameters
data_type
is the code number for the specified ODBC datatype about which information is returned. Datatype codes are listed in Table 2-2 on page 476 and Table 2-3 on page 476.

Usage
- The results set for sp_datatype_info is:

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.
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.

Permissions
Any user can execute sp_datatype_info.

sp_fkeys

Description

Returns information about foreign key constraints created with the create table or alter table command in the current database.

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. The use of wildcard characters in pattern matching is not supported. You must specify either the *pktable_name* or the *fktable_name*, or both.

pktable_owner
is the name of the primary key table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, sp_fkeys looks 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. The use of wildcard characters in pattern matching is not supported. Either the *fktable_name* or the *pktable_name*, or both, must be given.

fktable_owner
is the name of the foreign key table owner. The use of wildcard characters in pattern matching is not supported. If an *fktable_owner* is not specified, sp_fkeys looks 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.

Usage

- sp_fkeys returns information about foreign key constraints created with the create table or alter table command in the current database. A foreign key is a key column in a table that logically depends on a **primary key** column in another table.
- The results set for sp_fkeys is:

Column	Datatype	Description
pktable_qualifier	varchar(32)	The database that contains the primary key table.

Column	**Datatype**	**Description**
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 multicolumn 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.

- Both the primary key and foreign key must have been declared in a create table or alter table statement.
- If the primary key table name is supplied, but the foreign key table name is NULL, sp_fkeys 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, sp_fkeys returns all tables that are related by a primary key/foreign key relationship to foreign keys in the foreign key table.
- sp_fkeys does not return information about keys declared with sp_commonkey, sp_foreignkey or sp_primarykey.

Permissions

Any user can execute sp_fkeys.

sp_pkeys

Description

Returns information about primary key constraints created with the create table or alter table command for a single table.

Syntax

```
sp_pkeys table_name [, table_owner]
          [, table_qualifier]
```

Parameters

table_name
is the name of the table. The use of wildcard characters in pattern matching is not supported.

table_owner
is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If *table_owner* is not specified, sp_pkeys looks 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.

Usage

- The results set for sp_pkeys is:

Column	Datatype	Description
table_qualifier	varchar(32)	The database name. This field can be NULL.
table_owner	varchar(32)	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
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 multicolumn primary key.

- Primary keys must have been declared with the create table or alter table statement, not with sp_primarykey.
- The term **primary key** refers to a logical primary key for a table. Adaptive 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.

Permissions

Any user can execute sp_pkeys.

sp_server_info

Description
Returns a list of Adaptive Server attribute names and current values.

Syntax
sp_server_info [*attribute_id*]

Parameters
attribute_id
is the integer ID of the server attribute.

Examples
Example 1

```
sp_server_info 12

attribute_id attribute_name           attribute_value
------------ ------------------------ -------------------------
          12 MAX_OWNER_NAME_LENGTH  0
```

Example 2 Returns the list of server attributes, described by the mandatory rows, and their values:

```
sp_server_info
```

Usage
The results set for sp_server_info is:

Column	Datatype	Description
attribute_id	int	NOT NULL.
attribute_name	varchar(60)	NOT NULL.
attribute_value	varchar(255)	

The mandatory rows in the results set returned by sp_server_info are:

ID	Server attribute name	Description	Value
1	DBMS_NAME	Name of the DBMS.	`SQL SERVER`
2	DBMS_VER	Version of the DBMS.	*@@version*
6	DBE_NAME	Unused	
10	OWNER_TERM	Adaptive Server's term for a table owner (the second part of a three-part name).	`owner`
11	TABLE_TERM	Adaptive 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`
13	TABLE_LENGTH	The maximum number of characters for a table name.	`30`
14	MAX_QUAL_LENGTH	Maximum length of the name for a table qualifier (the first part of a three-part table name).	`30`

ID	Server attribute name	Description	Value
15	COLUMN_LENGTH	The maximum number of characters for a column name.	`30`
16	IDENTIFIER_CASE	The case sensitivity of user-defined names (table names, column names, and stored procedure names) in the database (the case in which these objects are presented in the system catalogs).	`MIXED`
18	COLLATION_SEQ	The assumed ordering of the character set for this server.	
19	SAVEPOINT_SUPPORT	Does the underlying DBMS support named savepoints?	`Y`
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`
22	ACCESSIBLE_TABLES	In sp_tables, 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`
100	USERID_LENGTH	The maximum number of characters for a user name.	`30`
101	QUALIFIER_TERM	Adaptive Server's term for a table qualifier (the first part of a three-part name).	`database`
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`
104	ACCESSIBLE_SPROC	In sp_stored_procedures, 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.	`32`
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?	`Y`
112	SP_RENAME	Can a stored procedure be renamed?	`Y`

ID	**Server attribute name**	**Description**	**Value**
500	SYS_SPROC_VERSION	The version of the catalog stored procedures currently implemented.	`01.01.2822`

Permissions Any user can execute sp_server_info.

sp_special_columns

Description

Returns the optimal set of columns that uniquely identify a row in a table or view; can also return a list of timestamp columns, whose values are automatically generated 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. The use of wildcard characters in pattern matching is not supported.

table_owner
is the name of the table or view owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, sp_special_columns looks 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 timestamp columns, whose values are generated by Adaptive Server each time a row is inserted or updated.

Examples

Example 1

```
sp_special_columns systypes

scope  column_name  data_type  type_name  precision  length  scale
------ ------------ ---------- ---------- ---------- ------- ------
   0   name          12        varchar         30       30   NULL
```

Example 2

```
sp_special_columns @table_name=authors, @col_type=R

scope  column_name  data_type  type_name  precision  length  scale
------ ------------ ---------- ---------- ---------- ------- ------
   0   au_id         12        varchar       11         11   NULL
```

Usage

- The results set for sp_special_columns is:

Column	Datatype	Description
scope	int	NOT NULL. Actual scope of the row ID. Adaptive Server always returns 0.

Column	Datatype	Description
column_name	varchar(30)	NOT NULL. Column identifier.
data_type	smallint	The integer code for an ODBC datatype. If this datatype cannot be mapped to an ANSI/ISO type, the value is NULL. The native datatype name is returned in the type_name column. (See the ODBC datatypes Table 2-2.)
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.

Permissions

Any user can execute sp_special_columns.

sp_sproc_columns

Description

Returns information about a stored procedure’s input and return parameters.

Syntax

sp_sproc_columns *procedure_name* [, *procedure_owner*]
[, *procedure_qualifier*] [, *column_name*]

Parameters

procedure_name
is the name of the stored procedure. The use of wildcard characters in pattern matching is not supported.

procedure_owner
is the owner of the stored procedure. The use of wildcard characters in pattern matching is not supported. If no owner is specified, sp_sproc_columns returns all columns.

procedure_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.

Usage

- The results set for sp_sproc_columns is:

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	
data_type	smallint	The integer code for an ODBC datatype. If this datatype cannot be mapped to an ANSI/ISO type, the value is 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.

Column	Datatype	Description
remarks	varchar(254)	NULL.
ss_data_type	tinyint	An Adaptive Server datatype.
colid	tinyint	An Adaptive Server specific column appended to the result set.

- sp_sproc_columns reports the type_name as float, and data_type as 6 for parameters defined as double precision. The Adaptive Server double precision datatype is a float implementation supports the range of values as specified in the ODBC specifications.

Permissions

Any user can execute sp_sproc_columns.

sp_statistics

Description

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. The use of wildcard character pattern matching is not supported.

table_owner
is the owner of the table. The use of wildcard character pattern matching is not supported. If *table_owner* is not specified, sp_statistics looks 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. The use of wildcard character pattern matching is not supported.

is_unique
is Y to return only unique indexes; otherwise, is N to return both unique and nonunique indexes.

Examples

```
sp_statistics publishers
```

```
table_qualifier                  table_owner
    table_name                       non_unique
    index_qualifier                  index_name
    type   seq_in_index column_name                    collation
    cardinality pages
-------------------------------- --------------------------------
   -------------------------------- ----------
   -------------------------------- ---------------------------
   ------ ------------ ------------------------------- --------
   ----------- -----------
pubs2                            dbo
    publishers                       NULL
    NULL                             NULL
        0         NULL NULL                            NULL
             3           1
pubs2                            dbo
    publishers                                0
    publishers                       pubind
```

```
            1              1 pub_id                             A
                 3             1
```

Usage

- The results set for sp_statistics is:

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.
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 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.

- The indexes in the results set appear in ascending order, ordered by the non-unique, type, index_name, and seq_in_index columns.
- The index type hashed accepts exact match or range searches, but searches involving pattern matching do not use the index.

Permissions

Any user can execute sp_statistics.

sp_stored_procedures

Description

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 wildcard characters to request information about more than one stored procedure.

sp_owner
is the owner of the stored procedure. Use wildcard characters to request information about procedures that are owned by more than one user.

sp_qualifier
is the name of the database. This can be the current database or NULL.

Usage

- sp_stored_procedures returns information about stored procedures in the current database only.
- The results set for sp_stored_procedures is:

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. Always returns -1.
num_output_params	int	NOT NULL. The value >= 0 shows the number of parameters; -1 means the number of parameters is indeterminate.
num_result_sets	int	NOT NULL. Always returns -1.
remarks	varchar(254)	NULL.

- sp_stored_procedures can return the name of stored procedures for which the current user does not have execute permission. However, 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.

Permissions

Any user can execute sp_stored_procedures.

sp_table_privileges

Description

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. The use of wildcard characters in pattern matching is not supported.

table_owner
is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, sp_table_privileges looks 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.

Usage

- The results set for sp_table_privileges is:

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.
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, and NULL.

Permissions

Any user can execute sp_table_privileges.

sp_tables

Description

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 wildcard characters to request information about more than one table.

table_owner
is the table owner. Use wildcard characters 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
is 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

```
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.

Usage

- Adaptive Server does not necessarily check the 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.
- 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.
- The results set for sp_tables is:

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

Permissions

Any user can execute sp_tables.

Tables used

master.dbo.sysattributes, master.dbo.sysloginroles, master.dbo.syssrvroles, sysroles

CHAPTER 3

System Extended Stored Procedures

This chapter describes the system extended stored procedures (ESPs), which are supplied by Sybase. ESPs are created by installmaster at installation. They are located in the sybsystemprocs database and owned by the System Administrator. They can be run from any database.

Topics covered are:

Topics	Page
Overview	499
Permissions on system ESPs	500
DLLs associated with system ESPs	500
Using system ESPs	500

Overview

Table 3-1 lists the system extended stored procedures discussed in this chapter.

Table 3-1: System extended stored procedures

Procedure	Description	Platform
xp_cmdshell	Executes a native operating system command on the host system running Adaptive Server.	All Supporting DLLs
xp_deletemail	Deletes a message from the Adaptive Server message inbox.	NT Only
xp_enumgroups	Displays groups for a specific Windows NT domain.	NT Only
xp_findnextmsg	Retrieves the message identifier of the next message in the Adaptive Server message inbox.	NT Only
xp_logevent	Provides for logging a user-defined event in the Windows NT Event Log.	NT Only
xp_readmail	Reads a message from the Adaptive Server message inbox.	NT Only
xp_sendmail	Sends a message to the specified recipients using the MAPI interface.	NT Only
xp_startmail	Starts an Adaptive Server mail session.	NT Only
xp_stopmail	Stops an Adaptive Server mail session.	NT Only

Permissions on system ESPs

Permissions are set in the sybsystemprocs database.

Users with the sa_role have default execution permissions on the system ESPs. These System Administrators can grant execution permissions to other users.

DLLs associated with system ESPs

You can get the names of the DLLs associated with the system ESPs by running sp_helpextendedproc in the sybsystemprocs database.

Using system ESPs

The system ESPs follow the same calling conventions as the regular system procedures. The only additional requirement for system ESPs is that the Open Server application, XP Server, must be running. Adaptive Server starts XP Server the first time an ESP is invoked. XP Server continues to run until you shut down Adaptive Server.

xp_cmdshell

Description

Executes a native operating system command on the host system running Adaptive Server.

Syntax

xp_cmdshell *command* [, no_output]

Parameters

command
is the operating system command string; maximum length is 255 bytes.

no_output
if specified, suppresses any output from the command.

Examples

Example 1 Silently copies the file named log on the C drive to a file named *log.0102* on the A drive:

```
xp_cmdshell 'copy C:\log A:\log.0102', no_output
```

Example 2 Executes the operating system's date command and returns the current date as a row of data:

```
xp_cmdshell 'date'
```

Usage

- xp_cmdshell returns any output, including operating system errors, as rows of text in a single column.
- xp_cmdshell is run from the current directory of the XP Server.
- The width of the column of returned output is 80 characters. The output is not formatted.
- xp_cmdshell cannot perform commands that require interaction with the user, such as "login".
- The user context in which an operating system command is executed via xp_cmdshell is controlled by the value of the xp_cmdshell context configuration parameter. If this parameter is set to 1 (the default), xp_cmdshell restricts permission to users with System Administration privileges at the operating system level. If this parameter is set to 0, xp_cmdshell uses the security context of the operating system account under which Adaptive Server is running. Therefore, using xp_cmdshell with the xp_cmdshell context configuration parameter set to 0, any user can execute operating system commands using the permissions of the account running Adaptive Server. This account may have fewer restrictions than the user's own account.

- Regardless of the value of xp_cmdshell context, if the user who is executing xp_cmdshell is not a System Administrator (does not have the sa_role), a System Administrator must have granted that user explicit permission to execute xp_cmdshell. For example, the following statement grants "joe" permission to execute xp_cmdshell:

  ```
  grant execute on xp_cmdshell to joe
  ```

- To find out if xp_cmdshell was successful in spawning an external command XP Server, enter the following, where *command* is the name of the command you ran with xp_cmdshell:

  ```
  @ret = exec xp_cmdshell command
  ```

  If xp_cmdshell was successful, @ret = exec xp_cmdshell *command* returns a value of 0. If xp_cmdshell failed, @ret = exec xp_cmdshell *command* returns a value of 1.

- To find out if the command you ran using xp_cmdshell was itself successful, enter the following, where *command* is the name of the command you ran with xp_cmdshell:

  ```
  @ret = exec xp_cmdshell command, return_status
  ```

  @ret = exec xp_cmdshell *command*, return_status causes xp_cmdshell to return the actual exit status code of the command. If a failure occurrs and XP Server cannot run the command, xp_cmdshell returns a value of 1. If the command runs successfully, xp_cmdshell returns a value of 0.

  If the command was successful, @ret = exec xp_cmdshell *command* returns a value of 0. If the command failed, @ret = exec xp_cmdshell *command* returns a value of 1.

  Note Both @ret = exec xp_cmdshell *command* and @ret = exec xp_cmdshell *command*, return_status are backward-compatible. Old stored procedures that do not use the return_status parameter treat @ret = exec xp_cmdshell *command*, return_status as if it were @ret = exec xp_cmdshell *command*.

  Also, the no_output parameter can still be used in combination with return_status, in any order.

Permissions

By default, only a System Administrator can execute xp_cmdshell. A System Administrator can grant execute permission to other users.

See also

See the *System Administration Guide* for more information about xp_cmdshell context.

xp_deletemail

Description

Windows NT only Deletes a message from the Adaptive Server message inbox.

Syntax

xp_deletemail [*msg_id*]

Parameters

msg_id
is the message identifier of the mail message to be deleted.

Examples

Example 1 Deletes from the Adaptive Server message inbox the message with the message identifier specified in the cur_msg_id variable:

```
1> declare @cur_msg_id binary(255)
2> exec xp_deletemail @msg_id = @cur_msg_id
```

Example 2 Deletes the first message from the Adaptive Server message inbox:

```
xp_deletemail
```

Usage

- Obtain the *msg_id* using xp_findnextmsg.
- If the *msg_id* parameter is not used, the message to be deleted defaults to the first message in the message inbox.

Permissions

By default, only a System Administrator can execute xp_deletemail. A System Administrator can grant this permission to other users.

xp_enumgroups

Description

Windows NT only Displays groups for a specified Windows NT domain.

Syntax

xp_enumgroups [*domain_name*]

Parameters

domain_name
is the Windows NT domain for which you are listing user groups.

Examples

Example 1 Lists all user groups on the Windows NT computer running XP Server:

```
xp_enumgroups
```

Example 2 Lists all user groups in the PCS domain:

```
xp_enumgroups 'PCS'
```

Usage

- xp_enumgroups displays all local user groups if no parameter is passed.
- A *domain* is a named collection of computers that share a common user account database and security policy.
- A return status of 0 indicates success; 1 indicates failure.

Permissions

By default, only a System Administrator can execute xp_enumgroups. A System Administrator can grant this permission to other users.

xp_findnextmsg

Description

Windows NT only Retrieves the next message identifier from the Adaptive Server message inbox.

Syntax

xp_findnextmsg @*msg_id* = @*msg_id* output [, *type*]
[, unread_only = {true | false}]

Parameters

msg_id
on input, specifies the message identifier that immediately precedes the one you are trying to retrieve. Places the retrieved message identifier in the *msg_id* output parameter, which must be of type binary.

type
is the input message type based on the MAPI mail definition. The only supported message type is CMC:IPM. A NULL value or no value defaults to CMC:IPM.

unread_only
if this parameter is set to true, xp_findnextmsg considers only unread messages. If this parameter is set to false, xp_findnextmsg considers all messages, both read and unread, when retrieving the next message identifier. The default is true.

Examples

Example 1 Returns, in the *@out_msg_id* output variable, the message identifier of the next unread message after the message specified by the *@out_msg_id*:

```
xp_findnextmsg @msg_id = @out_msg_id output
```

Example 2 Returns, in the *@out_msg_id* output variable, the message identifier of the next message after the message specified by the *@out_msg_id*. The message may be read or unread:

```
xp_findnextmsg @msg_id = @out_msg_id output, NULL,
@unread_only = false
```

Usage

- When xp_findnextmsg can find no more messages in the inbox, it returns a status of 1.
- xp_deletemail and xp_readmail use the message identifier returned by xp_findnextmsg.

Permissions

By default, only a System Administrator can execute xp_findnextmsg. A System Administrator can grant this permission to other users.

xp_logevent

Description

Windows NT only Provides for logging a user-defined event in the Windows NT Event Log from within Adaptive Server.

Syntax

xp_logevent *error_number*, *message* [, *type*]

Parameters

error_number
is the user-assigned error number. It must be equal to or greater than 50000.

message
is the text of the message that is displayed in the description field of the event viewer. The maximum length of the message is 255 bytes. Enclose the message in quotes.

type
describes the urgency of the event. Values are informational, warning, and error. The default is informational. Enclose the value in quotes.

Examples

Example 1 An informational event, number 55555, will be logged in the Windows NT Event Log. The text of the description in the event detail window is “Email message deleted”:

```
xp_logevent 55555, 'Email message deleted.'
```

Example 2 An error event, number 66666, will be logged in the Windows NT Event Log. The text of the description in the event detail window is “DLL not found”:

```
xp_logevent 66666, 'DLL not found.', 'error'
```

Usage

- The following table describes the default event details for events generated with xp_logevent:

Detail	Value
User	N/A
Computer	Name of machine running XP Server
Event ID	12
Source	Name of Adaptive Server
Category	User

Permissions

Only a System Administrator can execute xp_logevent.

xp_readmail

Description

Windows NT only Reads a message from the Adaptive Server message inbox.

Syntax

```
xp_readmail [msg_id ]
          [, recipients  output]
          [, sender  output]
          [, date_received  output]
          [, subject  output]
          [, cc  output]
          [, message  output]
          [, attachments  output]
          [, suppress_attach = {true | false}]
          [, peek = {true | false}]
          [, unread = {true | false}]
          [, msg_length  output]
          [, bytes_to_skip  [output]]
          [, type [output]]
```

Parameters

msg _id
specifies the message identifier of the message to be read by xp_readmail. If the *msg_id* parameter is not used, the message defaults to the first unread message in the message box, if unread is true, or to the first message in the message box, if unread is false.

recipients
is a semicolon-separated list of the recipients of the message.

sender
is the originator of the message.

date_received
is the date the message was received.

subject
is the subject header of the message.

cc
is a list of the message's copied (cc'd) recipients (separated by semicolons).

message
is the text of the message body. If the length of the message body, obtained from the *msg_length* output parameter, is greater than 255, use the *byte_to_skip* and *msg_length* parameters to read the message in 255-byte increments.

attachments
is a list of the temporary paths of the attachments (separated by semicolons). *attachments* is ignored if suppress_attach is true.

suppress_attach
if set to true, prevents the creation of temporary files for attachments. The default is true.

peek
if set to false, flags the message as unread after it has been read. If set to true, flags the message as an unread message, even after it has been read. The default is false.

unread_only
if set to true, xp_readmail considers only unread messages. If set to false, xp_readmail considers all messages, whether they are flagged as read or unread. The default is true.

msg_length
is the total length of the message, in bytes. Used with the *bytes_to_skip* parameter, allows xp_readmail to read messages in 255-byte increments.

bytes_to_skip
on input, if not 0, specifies the number of bytes to skip before reading the next 255 bytes of the message into the message output parameter. On output, contains the offset in the message (the previous value of *bytes_to_skip* plus the *msg_length* that is output with the call) from which to start reading the next 255-byte increment.

type
is the message type based on the MAPI mail definition. The only supported message type is CMC:IPM. A NULL value or no value defaults to CMC:IPM.

Examples

Example 1 xp_readmail reads the first unread message in the message inbox. It gets the message identifier for this message from the *@msgid* variable, where it has been stored by the xp_findnextmsg ESP. xp_readmail stores the sender's name in the *@originator* variable and the message body in the *@mess* variable:

```
declare @msgid binary(255)
declare @originator varchar(20)
declare @mess varchar(255)
exec xp_findnextmsg @msg_id = @msgid output
exec xp_readmail @msg_id = @msgid,
@sender = @originator output,
@message = @mess output
```

Example 2 Reads the first 255 bytes of the message for which the message identifier is output by xp_findnextmsg. If the total length of the message exceeds 255 bytes, reads the next 255 bytes and continues until there are no more bytes to read:

```
declare @msgid binary(255)
declare @mess varchar(255)
declare @msg_length char(255)
declare @len int
declare @skip int
exec xp_findnextmsg @msgid output
exec xp_readmail @msg_id = @msgid,
@message = @mess output,
@msg_length = @len output,
@bytes_to_skip = @skip output
print @mess
if (@len > 255)
begin
        while (@skip < @len)
        begin
            xp_readmail @msg_id = @msgid,
            @message = @mess output,
            @bytes_to_skip = @skip output
            print @mess
        end
end
```

Usage

- xp_readmail reads a message from the Adaptive Server message inbox.
- To get the message identifier of the next message in the message inbox, use xp_findnextmsg.

Permissions

By default, only a System Administrator can execute xp_readmail. A System Administrator can grant this permission to other users.

xp_sendmail

Description

Windows NT only Sends a message to the specified recipients. The message is either text or the results of a Transact-SQL query.

Syntax

```
xp_sendmail recipient [; recipient] . . .
          [, subject]
          [, cc_recipient] . . .
          [, bcc_recipient] . . .
          [, {query | message}]
          [, attachname]
          [, attach_result = {true | false}]
          [, echo_error = {true | false}]
          [, include_file [, include_file] . . .]
          [, no_column_header = {true | false}]
          [, no_output = {true | false}]
          [, width]
          [, separator]
          [, dbuser]
          [, dbname]
          [, type]
          [, include_query = {true | false}]
```

Parameters

recipient
is the email address of the user who will receive the message. At least one recipient is required. Separate multiple recipients with semicolons.

subject
is the optional message subject header. If not used, defaults to "Sybase SQL Server Message".

cc_recipient
is a list of the message's copied (cc'd) recipients (separated by semicolons).

bcc_recipient
is the list of the message's blind- copied (bcc'd) recipients (separated by semicolons).

query
is one or more Transact-SQL statements. The results are sent to the recipients of the message. If *query* is used, *message* cannot be used.

message
is the text of the message being sent. If *message* is used, *query* cannot be used. For the complete list of options that are ignored when you use message, see the "Usage" section.

attachname

is the name of the file containing the results of a query, which is included as an attachment to the message, when the *query* parameter is used. If *attachname* is used, *attach_result* must be set to true. If *attach_result* is true and *attachname* is not specified, the prefix of the attached file's generated file name is "syb" followed by 5 random digits followed by the ".txt" extension, for example, *syb84840.txt*. This parameter is ignored if the *message* parameter is used.

attach_result

if set to true, sends the results of a query as an attachment to the message. If set to false, sends the results directly in the message body. The default is false. This parameter is ignored if the *message* parameter is used.

echo_error

if set to true, sends Adaptive Server messages, including the count of rows affected message, along with the query results. If set to false, does not send Adaptive Server messages. The default is true. This parameter is ignored if the *message* parameter is used.

include_file

is a list of files to be included as attachments to the message, separated by semicolons. The files can be specified as file names, path names, or relative path names and can be either text or binary files.

no_column_header

if set to true, column headers are sent with query results. If set to false, column headers are not sent. The default is false. This parameter is ignored if the *message* parameter is used.

no_output

if set to true, no output is sent to the session that sent the mail. If set to false, the session sending the mail receives output. The default is false. This parameter is ignored if the *message* parameter is used.

width

specifies, in characters, the width of the results sets when query results are sent in a message. *width* is the same as the /w option in isql. Result rows are broken by the newline character when the specified *width* is reached. The default is 80 characters. This parameter is ignored if the *message* parameter is used.

separator

specifies the character to be used as a column separator when query results are sent in a message. *separator* is the same as the /s option in isql. The default is the tab character. This parameter is ignored if the *message* parameter is used.

dbuser

specifies the database user name to be assumed for the user context for executing queries when the *query* parameter is used. The default is "guest." This parameter is ignored if the *message* parameter is used.

dname

specifies the database name to be assumed for the database context for executing queries when the *query* parameter is used. The default is "master." This parameter is ignored if the *message* parameter is used.

type

is the input message type based on the MAPI mail definition. The only supported message type is CMC:IPM. A NULL value or no value defaults to CMC:IPM.

include_query

if set to true, the query or queries used in the *query* parameter are appended to the results set. If set to false, the query is not appended. The default is false. *include_query* is ignored if the *message* parameter is used.

Examples

Example 1 xp_sendmail sends a text message on the backup status of an Adaptive Server to "sally" and "ramon" with a copy to the "admin" group:

```
xp_sendmail @recipient = "sally;ramon",
@subject = "Adaptive Server Backup Status",
@message = "Adaptive Server Backup for SERVER2 is
complete.",
@copy_recipient="admin"
```

Example 2 Sends "peter" the results of a query on the *authors* table. The results are in an attachment to the message, which consists of a file named *au_lis.res*, which is in the directory from which the server is being executed:

```
xp_sendmail "peter",
@query = "select * from authors",
@attachname = "au_list.res",
@attach_result= true
```

Usage

- The following parameters are related to the results of queries sent in a message when the query parameter is used. They are ignored if the message parameter is used instead: attachname, attach_result, echo_error, no_column_header, no_output, width, separator, dbuser, dname, include_query.

Permissions

By default, only a System Administrator can execute xp_sendmail. A System Administrator can grant this permission to other users.

xp_startmail

Description

Windows NT only Starts an Adaptive Server mail session.

Syntax

xp_startmail [*mail_user*] [, *mail_password*]

Parameters

mail_user
is a mail profile name used by Adaptive Server to log into the Windows NT mail system. If *mail_user* is not used, xp_startmail uses the mail user name that was used to set up Sybmail's Adaptive Server account.

mail_password
is the mail password used by Adaptive Server to log into the Windows NT mail system. If *mail_password* is not used, xp_startmail uses the mail password that was used to set up Sybmail's Adaptive Server account.

Examples

Example 1 Starts an Adaptive Server mail session using the mail user name and password for Sybmail's user account:

```
xp_startmail
```

Example 2 Starts an Adaptive Server mail session with "mailuser" as the profile name and the password associated with that profile name:

```
xp_startmail "mailuser", "tre55uu"
```

Usage

- xp_startmail will not start an Adaptive Server mail session if one is already running.
- An Adaptive Server mail session must be started, either by an explicit call to xp_startmail or by configuring Adaptive Server to start an Adaptive Server mail session automatically at start-up, before any Sybmail-related system ESPs or the sp_processmail stored procedure can be executed. See start mail session in the *System Administration Guide* for information about initiating an Adaptive Server mail session automatically at start-up.
- When the Windows NT automail session is not on, you must use the *mail_user* and *mail_password* parameters with xp_startmail.
- To see the default *mail_user* value from the *fullname* field for the "sybmail" user account, use the sp_displaylogin system procedure as follows:

```
sp_displaylogin sybmail
```

Permissions

By default, only a System Administrator can execute xp_startmail. A System Administrator can grant this permission to other users.

xp_stopmail

Description	**Windows NT only** Stops an Adaptive Server mail session.
Syntax	xp_stopmail
Parameters	None
Examples	Stops an Adaptive Server mail session:

```
xp_stopmail
```

Usage

- Sybmail-related system ESPs and the sp_processmail stored procedure cannot be executed after an Adaptive Server mail session has been terminated with xp_stopmail.

Permissions

By default, only a System Administrator can execute xp_stopmail. A System Administrator can grant this permission to other users.

CHAPTER 4 *dbcc* Stored Procedures

This chapter describes the dbcc stored procedures.

Topics covered are:

Topics	Page
Overview	517
Specifying the object name and date	518

Overview

These procedures access the tables only in the dbccdb database or in the alternate database, dbccalt. See the *System Administration Guide* for details on setting up dbccdb or dbccalt. See Chapter 2, "dbccdb Tables" in *Reference Manual: Tables* for information on the tables used in these databases.

Table 4-1 lists the dbcc stored procedures described in this chapter. For details on the dbcc system procedure sp_plan_dbccdb, see sp_plan_dbccdb. See the *System Administration Guide* for more information on this system procedure and the dbcc stored procedures.

Table 4-1: dbcc stored procedures

Procedure name	Description
sp_dbcc_alterws	Changes the size of the specified workspace to a specified value, and initializes the workspace.
sp_dbcc_configreport	Generates a report that describes the configuration information used by the dbcc checkstorage operation for the specified database.
sp_dbcc_createws	Creates a workspace of the specified type and size on the specified segment and database.
sp_dbcc_deletedb	Deletes from dbccdb all the information related to the specified target database.
sp_dbcc_deletehistory	Deletes the results of dbcc checkstorage operations performed on the target database before the specified date and time.
sp_dbcc_differentialreport	Generates a report that highlights the changes in I/O statistics and faults that took place between two dbcc operations

Procedure name	Description
sp_dbcc_evaluatedb	Recomputes configuration information for the target database and compares it to the current configuration information.
sp_dbcc_faultreport	Generates a report covering fault statistics for the dbcc checkstorage operations performed for the specified object in the target database on the specified date, listed in order by table and index.
sp_dbcc_fullreport	Runs sp_dbcc_summaryreport, sp_dbcc_configreport, sp_dbcc_statisticsreport, and sp_dbcc_faultreport.
sp_dbcc_recommendations	Prints recommendations for a database based on a checkstorage run and groups the recommendations by table and index.
sp_dbcc_runcheck	Runs dbcc checkstorage on the specified database, then runs sp_dbcc_summaryreport or a report you specify.
sp_dbcc_statisticsreport	Generates an allocation statistics report on the specified object in the target database.
sp_dbcc_summaryreport	Generates a summary report on the specified database.
sp_dbcc_updateconfig	Updates the dbcc_config table in dbccdb with the configuration information of the target database.

Specifying the object name and date

Several dbcc stored procedures use parameters for the object name and date. This section provides important information on specifying the object name and date.

Specifying the object name

The object name specifies only the name of the table or index for which to generate a report. When you specify an object name, you must also specify a database name (*dbname*). You cannot specify an owner for the object. If the specified object name is not unique in the target database, the system procedure generates a report on all objects with the specified name.

Specifying the date

Use the following syntax to specify the date and time (optional):

```
mm/dd/yy[:hh:mm:ss]
```

A 24-hour clock is assumed.

When you specify the date, the system procedures interpret it as follows:

- If both the date and the time are specified, the dbcc operation that completed at the specified date and time is selected for the report.
- If the specified date is the current date, and no time is specified, the time is automatically set to the current time. The dbcc operation that completed within the previous 24 hours with a finish time closest to the current time is selected for the report.
- If the specified date is not the current date, and no time is specified, the time is automatically set to "23:59:59". The dbcc checkstorage operation that completed with a finish date and time closest to the specified date and system-supplied time is selected for the report.

For example, suppose the most recent dbcc checkstorage operation completed on March 4, 1997 at 10:20:45.

If you specify the date as "03/04/97", the system procedure interprets the date as 03/04/97:23:59:59. This date and time are compared to the actual finish date and time of 03/04/97:10:20:45.

If you specify the date as "03/04/97:10:00:00", the operation that completes at 10:20:45 is not selected for the report because only the operations that complete on or before the specified time meet the criteria.

If you specify the date as "03/06/97", no report is generated because the most recent operation completed more than 24 hours earlier.

sp_dbcc_alterws

Description

Changes the size of the specified workspace to a specified value, and initializes the workspace.

Syntax

sp_dbcc_alterws *dbname*, *wsname*, "*wssize*[K|M]"

Parameters

dbname
is the name of the database in which the workspace resides. Specify either dbccdb and dbccalt.

wsname
specifies the name of the workspace to alter.

wssize
is the new size of the workspace, specified by K (kilobytes) or M (megabytes). If you do not specify K or M, *wssize* specifies the number of pages. Page size is platform-dependent. The minimum size for a workspace is 24 pages.

Examples

Changes the size of the scan_ws_000001 workspace on dbccdb to 30MB:

```
sp_dbcc_alterws dbccdb, scan_ws_000001, "30M"
Workspace scan_ws_000001 has been altered successfully to size 30MB
```

Usage

- sp_dbcc_alterws changes the size of the specified workspace to the specified value and initializes the workspace.
- To achieve maximum performance, make sure you have configured a buffer pool of at least 16K before you alter a workspace.
- Use sp_plan_dbccdb to determine size estimates before altering the workspace.
- The workspace must exist before it can be altered. For information on creating workspaces, see sp_dbcc_createws.
- To delete a workspace, in dbccdb issue:

```
drop table workspace_name
```

Permissions

Only a System Administrator or the Database Owner can run sp_dbcc_alterws.

See also

See the *System Administration Guide* for more information on the scan and text workspaces, and the dbccalt database.

Commands dbcc

dbcc stored procedures sp_dbcc_createws, sp_dbcc_evaluatedb

System procedures sp_plan_dbccdb, sp_helpdb

sp_dbcc_configreport

Description
Generates a report that describes the configuration information used by the dbcc checkstorage operation for the specified database.

Syntax
sp_dbcc_configreport [*dbname*]

Parameters
dbname
specifies the name of the database. If *dbname* is not specified, the report contains information on all databases in dbccdb..dbcc_operation_log.

Examples
Generates a report on the configuration information related to dbcc for the sybsystemprocs database. The "Value" column lists the object name, where applicable, and the size:

```
sp_dbcc_configreport

Reporting configuration information of database sybsystemprocs.

Parameter Name              Value                         Size

database name               sybsystemprocs                51200K
dbcc named cache            default data cache            1024K
text workspace              textws_001 (id = 544004969)   128K
scan workspace              scanws_001 (id = 512004855)   1024K
max worker processes        1
operation sequence number    2
```

Usage

- sp_dbcc_configreport generates a report that describes the configuration information used by dbcc operations for the specified database. This information is stored in the dbcc_config table.
- To evaluate the most current configuration parameters, run sp_dbcc_updateconfig before running sp_dbcc_configreport.
- To change the configuration values for a workspace, use sp_dbcc_alterws.

Permissions
Any valid user for the database name specified can run sp_dbcc_configreport.

database name specified.

See also
Commands dbcc

dbcc stored procedures sp_dbcc_alterws, sp_dbcc_fullreport, sp_dbcc_statisticsreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_createws

Description

Creates a workspace of the specified type and size on the specified segment and database.

Syntax

sp_dbcc_createws *dbname*, *segname*, [*wsname*], *wstype*, "wssize[K|M]"

Parameters

dbname
is the name of the database in which the workspace is to be created. Values are dbccdb and dbccalt.

segname
is the name of the segment for the workspace.

wsname
is the name of the workspace. If the value is null, sp_dbcc_createws generates the name scan_ws_nnnnnn for the scan workspace and text_ws_nnnnnn for the text workspace, where *nnnnnn* is a unique 6-digit number.

wstype
specifies the type of workspace to be create. Values are scan and text.

wssize
is the workspace size, specified with K (kilobytes) or M (megabytes). If you do not specify K or M, *wssize* specifies the number of pages. The minimum size for a workspace is 24 pages.

Examples

Example 1 Creates a 10MB scan workspace named scan_ws_pubs2 on the scanseg segment in dbccdb:

```
sp_dbcc_createws dbccdb, scanseg, scan_ws_pubs2, scan, "10M"
```

Example 2 Creates a 14MB scan workspace named text_ws_000001 on the textseg segment in dbccdb:

```
sp_dbcc_createws dbccdb, textseg, text, "14M"
```

Usage

- sp_dbcc_createws creates a workspace with the specified name and size and initializes it.
- Before you create a workspace, create the segment with sp_addsegment.
- Before you create a workspace, make sure you have configured a buffer pool of at least 16K, to achieve maximum performance.
- When you create a workspace, make sure to add a 5% overhead on the space needed on the device because of large page allocation scheme used when creating the workspace.
- Use sp_plan_dbccdb to determine size estimates.

- After creating a workspace, run sp_dbcc_updateconfig to record the new configuration information in dbcc_config.
- Each workspace must have a unique name.
- To delete a workspace, in dbccdb issue:

```
drop table workspace_name
```

Permissions

Only a System Administrator or the Database Owner can run sp_dbcc_createws.

See also

See the *System Administration Guide* for more information on the scan and text workspaces, and the dbccalt database.

Commands dbcc

dbcc stored procedures sp_dbcc_alterws, sp_dbcc_evaluatedb

System procedures sp_addsegment, sp_plan_dbccdb, sp_helpsegment

sp_dbcc_deletedb

Description

Deletes from dbccdb all the information related to the specified target database.

Syntax

sp_dbcc_deletedb [*dbname* | *dbid*]

Parameters

dbname
specifies the name of the target database for which you want the configuration information deleted. If you do not specify a value for *dbname*, Adaptive Server deletes data from all databases in dbccdb..dbcc_config. If the target database is dbccdb, and dbccalt exists, Adaptive Server deletes the data from dbccalt.

dbid
specifies the database ID number of the target database for which you want the configuration information deleted.

Examples

Deletes all information for the database named engdb from dbccdb:

```
sp_dbcc_deletedb "engdb"
```

```
All information for database engdb has been deleted from dbccdb.
```

Usage

- sp_dbcc_deletedb deletes from dbccdb all the information related to the specified target database, including configuration information and the results of previous dbcc checkstorage operations.
- If the deleted database is dbccdb, and the dbccalt database exists, sp_dbcc_deletedb deletes the configuration information and results of dbccdb from dbccalt.
- To remove the results of dbcc checkstorage operations created before a specific date, use sp_dbcc_deletehistory.
- Using the *dbid* option is the only way to delete the contents of the dbccdb database for a database that has already been dropped.

Permissions

Only a System Administrator or the Database Owner can run sp_dbcc_deletedb.

See also

See the *System Administration Guide* for information about the dbccalt database.

Commands dbcc

dbcc stored procedures sp_dbcc_deletehistory, sp_dbcc_evaluatedb

System procedures sp_plan_dbccdb

sp_dbcc_deletehistory

Description

Deletes the results of dbcc checkstorage operations performed on the target database before the specified date and time.

Syntax

sp_dbcc_deletehistory [*cutoffdate* [, *dbname* | *dbid*]]

Parameters

cutoffdate
deletes all entries made on or before this date. This parameter is of type datetime. If a date is not specified, only the results of the last operation are retained. For more information, see “Specifying the date” on page 518.

dbname
specifies the name of the database for which the data must be deleted. If not specified, sp_dbcc_deletehistory deletes the history information for all databases in dbccdb..dbcc_config.

dbid
specifies the database ID number of the target database for which you want the history information deleted.

Examples

Deletes results of all operations performed on the database pubs2 on or before March 4, 1997:

```
sp_dbcc_deletehistory "03/04/1997", "pubs2"
```

Usage

- sp_dbcc_deletehistory deletes the results of dbcc checkstorage operations performed on the target database before the specified date and time.
- If the target database is dbccdb, and the dbccalt database exists, sp_dbcc_deletehistory deletes historical data for dbccdb from dbccalt.
- The value specified for *cutoffdate* is compared to the finish time of each dbcc operation.
- Use the *dbid* option to delete the historical data of the dbccdb database for a database that has already been dropped.
- Using the *dbid* option is the only way to delete the historical data of the dbccdb database for a database that has already been dropped.
- To see the dates when dbcc checkstorage was run so that you can choose the value for *cutoffdate*, run sp_dbcc_summaryreport.

Permissions

- Only a System Administrator or the Database Owner can run sp_dbcc_deletehistory on a specific database.
- Only a System Administrator can run sp_dbcc_deletehistory without specifying a database name.

See also

See the *System Administration Guide* for information on the dbccalt database.

Commands dbcc

dbcc stored procedures sp_dbcc_deletedb, sp_dbcc_evaluatedb

System procedures sp_plan_dbccdb

sp_dbcc_differentialreport

Description

Generates a report that highlights the changes in I/O statistics and faults that took place between two dbcc operations.

Syntax

sp_dbcc_differentialreport [*dbname* [, *objectname*]],
    [*db_op*] [, "*date1"* [, "*date2"*]]

Parameters

dbname
specifies the name of the database. If you do not specify a *dbname*, the report contains information on all databases in dbccdb..dbcc_operation_log.

objectname
specifies the name of the table or index for which you want the report generated. If *object_name* is not specified, statistics on all objects in the target database are reported.

db_op
specifies the source of the data to be used for the report. The only value is checkstorage. The report is generated on the data specified by *db_op* on *date1* and *date2* for the specified object in the target database. If dates are not specified, the last two operations of the type *db_op* are compared.

date1
specifies the first date of a dbcc checkstorage operation to be compared.

date2
specifies the last date of a dbcc checkstorage operation to be compared.

Examples

```
sp_dbcc_differentialreport master, sysprocedures,
checkstorage, "05/01/97", "05/04/97"
```

Generates a report that shows the changes in I/O statistics and faults that occurred in the sysprocedures table between May 1, 1997 and May 4, 1997

Usage

- sp_dbcc_differentialreport generates a report that highlights the changes in I/O statistics and faults that occurred between two dbcc operations. It compares counter values reported from two instances of dbcc checkstorage. Only the values that have been changed are reported.
- If only one date is specified, the results of the dbcc checkstorage operation selected by the specified date are compared to the results of the dbcc checkstorage operation immediately preceding the selected operation.
- If no dates are specified, the results of last two dbcc checkstorage operations are compared.
- If sp_dbcc_differentialreport returns a number for *object_name*, it means the object was dropped after the dbcc checkstorage operation completed.

- If no changes occurred between the specified operations, sp_dbcc_differentialreport does not generate a report.

Permissions

Any valid user for the database name specified can run sp_dbcc_differentialreport.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_fullreport, sp_dbcc_statisticsreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_evaluatedb

Description

Recomputes configuration information for the target database and compares it to the current configuration information.

Syntax

sp_dbcc_evaluatedb [*dbname*]

Parameters

dbname

specifies the name of the target database. If *dbname* is not specified, sp_dbcc_evaluatedb compares all databases listed in the dbcc_config table.

Examples

Recomputes configuration information for the current database, sybsystemprocs, and suggests new values for some parameters:

```
1> sp_dbcc_evaluatedb
2> go
```

```
Recommended values for workspace size, cache size and process count are:

Database name : one_G
                                      current          suggested
scan workspace size :                    750M                16M
text workspace size :                      2K                48K
cache size          :                  10240K              1280K
process count       :                       3                  2

Each of the reported quantities is reported in a scaled unit according to
G if size > 10G
M if 10M < size <=10 G
K otherwise
```

Usage

- sp_dbcc_evaluatedb recomputes configuration information for the target database and compares the data to the current configuration information. It uses counter values recorded for the target database in the dbcc_counters table.
- The cache size is the size of the 16K buffer pool in the cache. For a 2K buffer pool, the minimum size of this cache must be the recommended value, plus 512.
- When the size and data distribution pattern of the target database changes, run sp_dbcc_evaluatedb to optimize the configuration information.
- To gather configuration information for the target database the first time, use sp_plan_dbccdb.

- To make sure you are evaluating the most current configuration parameters, run sp_dbcc_updateconfig before running sp_dbcc_evaluatedb.

Permissions

- Only System Administrator or the Database Owner can run sp_dbcc_evaluatedb.
- Only a System Administrator can run sp_dbcc_evaluatedb without specifying a database name.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_updateconfig

System procedures sp_plan_dbccdb

sp_dbcc_faultreport

Description

Generates a report covering fault statistics for the dbcc checkstorage operations performed for the specified object in the target database on the specified date.The report lists the tables and indexes in order.

Syntax

sp_dbcc_faultreport [*report_type* [, *dbname*
[, *objectname* [, *date*], @hard_only]]]]

Parameters

report_type

specifies the type of fault report. Valid values are short and long. The default is short.

dbname

specifies the name of the target database; for example, master..sysdatabases. If *dbname* is not specified, the report contains information on all databases in dbccdb..dbcc_operation_log.

object_name

specifies the name of the table or index for which you want the report generated. If *object_name* is not specified, statistics on all objects in the target database are reported.

date

specifies exact date and time that the dbcc checkstorage operation finished. You can find this value in dbcc_operation_log.finish. You can create the value by combining the date from start time and the hours and minutes from end time in the sp_dbcc_summaryreport output. If you do not specify *date*, Adaptive Server uses the date of the most recent operation.

When you specify the *date* parameter, be certain that the time you enter is later than the date of the operation. sp_dbcc_faultreport cannot report faults that occur later than the time you enter in this parameter.

Note To focus on the *date* parameter, use "null" for all other parameters. If you omit a parameter entirely, sp_dbc_faultreport cannot generate a correct report.

Examples

Example 1 Generates a short report of the faults found in tables in the sybsystemprocs database. The report includes the table name, the index number in which the fault occurred, the type code of the fault, a brief description of the fault, and the page number on which the fault occurred:

```
sp_dbcc_faultreport "short"

Database Name : sybsystemprocs

 Table Name     Index  Type Code Description          Page Number
 ------------- ------ --------- ------------------ -----------
 sysprocedures        0    100031 page not allocated        5702
 sysprocedures        1    100031 page not allocated       14151
 syslogs              0    100022 chain start error        24315
 syslogs              0    100031 page not allocated       24315
```

Example 2 Generates a long report of the faults found in tables in the sybsystemprocs database. This example shows the first part of the output of a long report. The complete report repeats the information for each object in the **target database** in which dbcc checkstorage found a fault. The data following the long string of numbers shown under the "page header" field ("Header for 14151, next 14216, previous 14150 ...") describes the components of the "page header" string:

```
sp_dbcc_faultreport "long"
```

```
Generating 'Fault Report' for object sysprocedures in database
sybsystemprocs.

Type Code: 100031; Soft fault, possibly spurious
Page reached by the chain is not allocated.
page id: 14151
page header:
0x00003747000037880000374600000005000648B803EF0001000103FE0080000F
Header for 14151, next 14216, previous 14150, id = 5:1
 time stamp = 0x0001000648B8, next row = 1007, level = 0
 free offset = 1022, minlen = 15, status = 128(0x0080)
.
.
.
```

Example 3 Generates a short report of faults from all tables on all databases, for an operation finished at a date and time found as an End Time, from the output of sp_dbcc_summaryreport. It is important that you use accurate end times in the *date* parameter; for instance, if you enter:

```
7/25/2000 9:58
```

instead of

```
7/25/2000 9:58:0:190
```

the report will generate faults only up to 9:58, not after it. You could use 9:59 if you do not want to enter the exact time the operation ends:

```
sp_dbcc_faultreport "short", NULL, NULL,
   "07/25/00 9:59"
```

In this case the report will generate faults up to 9:59.

Usage

- sp_dbcc_faultreport generates a report that shows all faults for the specified object in the target database.
- sp_dbcc_faultreport issues numerous error message number 10028 If you use:
  - sp_placeobject to make an object that has existing allocations put new allocations on a new segment.
  - sp_dropsegment to remove a segment from a fragment that contains allocations of an object assigned to that segment.

    Error message number 100028 is an informational message rather than an indication of a serious error. If you prefer not to receive such messages, you can create your own reporting procedure that does not report this (or any other) error. One way to do this is to add the following to the very beginning of the standard sp_dbcc_faultreport stored procedure in the *installdbccdb* script:

```
print "removing 100028 errors from dbcc_faults table"
delete dbcc_faults where type_code = 100028
```

- If sp_dbcc_faultreport returns a number for *object_name*, it means the object was dropped after the dbcc checkstorage operation completed.

Permissions

Any valid user for the database name specified can run sp_dbcc_faultreport.

See also

See the type_code column described in the *System Administration Guide* for information on the fault ID and on the fault status.

Commands dbcc

dbcc stored procedures sp_dbcc_fullreport, sp_dbcc_statisticsreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_fullreport

Description

Runs sp_dbcc_summaryreport, sp_dbcc_configreport, sp_dbcc_statisticsreport, and sp_dbcc_faultreport short for *database..object_name* on or before the specified *date*.

Syntax

sp_dbcc_fullreport [*dbname* [, *objectname* [, *date*]]]

Parameters

dbname

specifies the name of the database. For example, master..sysdatabases. If you do not specify *dbname,* the report contains information on all databases in dbccdb..dbcc_operation_log.

object_name

specifies the name of the table or index for which you want the report generated. If you do not specify *object_name*, statistics on all objects in the target database are reported.

date

specifies the date on which the dbcc checkstorage operation was performed. If you do not specify a *date*, the date of the last operation is used.

Examples

Runs sp_dbcc_summaryreport, sp_dbcc_configreport, sp_dbcc_statisticsreport, and sp_dbcc_faultreport short for the most recent dbcc checkstorage operation run on the sysprocedures table in the master database:

```
sp_dbcc_fullreport master, sysprocedures
```

Usage

- sp_dbcc_fullreport runs sp_dbcc_summaryreport, sp_dbcc_configreport, sp_dbcc_statisticsreport, and sp_dbcc_faultreport short for database..object_name on or before the specified date

Permissions

Any valid user for the database name specified can run sp_dbcc_fullreport.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_statisticsreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_recommendations

Description

Reports recommendations for a database based on a checkstorage run grouping the recommendations by table and index.

Syntax

sp_dbcc_recommendations dbname [, *date* [, *opid* [, *objectname*]]]

Parameters

date
specifies the date on which the dbcc checkstorage operation was performed. If you do not specify a *date*, the date of the last checkstorage run is used.

opid
identifies the dbcc operation that was performed.

objectname
specifies the name of the table or index for which you want the report generated. If you do not specify *objectname*, statistics on all objects in the target database are reported.

Examples

Run the sp_dbcc_recommendations report on the most recent checkstorage run.

```
sp_dbcc_recommendations pubs2
```

If a date or opid are not used the report gives information on all of the latest information on the most recent checkstoragerun.

If a date and opid are used, the date is ignored

Usage

When the sp_dbcc_summaryreport is called with an optional date and the optional name is NULL or checkstorage, the sp_dbcc_recommendations is invoked at the end of the report.

The report lists the recommendations in a group for each table and index.

Permissions

Any valid user for the database name specified.

See also

sp_dbcc_summaryreport, checkstorage, sp_dbcc_faultreport.

sp_dbcc_runcheck

Description

Runs dbcc checkstorage on the specified database, then runs sp_dbcc_summaryreport or a report you specify.

Syntax

sp_dbcc_runcheck *dbname* [, *user_proc*]

Parameters

dbname
specifies the name of the database on which the check is to be performed.

user_proc
specifies the name of the dbcc stored procedure or a user-created stored procedure that is to be run instead of sp_dbcc_summaryreport.

Examples

Example 1 Checks the database engdb and generates a summary report on the information found:

```
sp_dbcc_runcheck "engdb"
```

Example 2 Checks the database pubs2 and generates a full report:

```
sp_dbcc_runcheck "pubs2", sp_dbcc_fullreport
```

Usage

- sp_dbcc_runcheck runs dbcc checkstorage on the specified database.
- After the dbcc checkstorage operation is complete, sp_dbcc_runcheck runs sp_dbcc_summaryreport to generate a summary report. If you specify one of the other report-generating dbcc stored procedures for *dbcc_report*, sp_dbcc_runcheck runs that procedure instead of sp_dbcc_summaryreport. See the *System Administration Guide* for a brief description and examples of all the report-generating stored procedures provided with dbccdb.
- You can write your own report-generating stored procedure and specify its name for user_proc. The stored procedure must be self-contained. sp_dbcc_runcheck cannot pass any parameters to Adaptive Server.

Permissions

Only a System Administrator or the Database Owner can run sp_dbcc_runcheck.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_summaryreport

sp_dbcc_statisticsreport

Description

Generates an allocation statistics report on the specified object in the target database.

Syntax

sp_dbcc_statisticsreport [*dbname* [, *objectname*
[, *date*]]]

Parameters

dbname

specifies the **target database**. If *dbname* is not specified, the report contains information on all databases in dbccdb..dbcc_operation_log.

objectname

specifies the name of the table or index for which you want the report generated. If you do not specify *objectname*, Adaptive Server reports statistics on all objects in the target database.

date

specifies the date on which the dbcc checkstorage operation was performed. If you do not specify *date*, Adaptive Server uses the date of the most recent operation.

Examples

Generates a statistics report on the sysobjects table in the sybsystemprocs database:

```
sp_dbcc_statisticsreport 'sybsystemprocs', 'sysobjects'

Statistics Report on object sysobjects in database sybsystemprocs

 Parameter Name            Index Id Value
 ------------------------- -------- ------------
 count                     0        241.0
 max size                  0        99.0
 max count                 0        22.0
 bytes data                0        19180.0
 bytes used                0        22113.0
 count                     1        14.0
 max size                  1        9.0
 max level                 1        0.0
 max count                 1        14.0
 bytes data                1        56.0
 bytes used                1        158.0
 count                     2        245.0
 max level                 2        1.0
 max size                  2        39.0
 max count                 2        71.0
 bytes data                2        4377.0
 bytes used                2        6995.0
 Parameter Name  Index Id  Partition  Value   Dev_name
---------------  --------  ---------  ------  -------------
page gaps               0          1    13.0  master
pages used              0          1    15.0  master
extents used            0          1     3.0  master
overflow pages          0          1     0.0  master
pages overhead          0          1     1.0  master
pages reserved          0          1     7.0  master
page extent gaps        0          1    11.0  master
ws buffer crosses       0          1     2.0  master
page extent crosses     0          1    11.0  master
pages used              1          1     2.0  master
extents used            1          1     1.0  master
overflow pages          1          1     0.0  master
pages overhead          1          1     1.0  master
pages reserved          1          1     6.0  master
page extent gaps        1          1     0.0  master
ws buffer crosses       1          1     0.0  master
page extent crosses     1          1     0.0  master
page gaps               2          1     4.0  master
pages used              2          1     6.0  master
extents used            2          1     1.0  master
overflow pages          2          1     0.0  master
pages overhead          2          1     1.0  master
pages reserved          2          1     2.0  master
page extent gaps        2          1     0.0  master
ws buffer crosses       2          1     0.0  master
page extent crosses      2         1     0.0  master
```

Usage

- sp_dbcc_statisticsreport generates an allocation statistics report on the specified object in the target database. It uses data from the dbcc_counters table, which stores information about page utilization and error statistics for every object in the target database.
- If sp_dbcc_statisticsreport returns a number for *object_name*, it means the object was dropped after the dbcc checkstorage operation completed.
- sp_dbcc_statisticsreport reports values recorded in the dbcc_counters table for the datatypes 5000–5024 See the System Administration Guide.

  For bytes data, bytes used, and overflow pages, sp_dbcc_statisticsreport reports the sum of the values reported for all partitions and devices.

  For count, max count, max size and max level, sp_dbcc_statisticsreport reports the largest of the values reported for all partitions and devices.

sp_dbcc_statisticsreport reports information for each device and partition used by objects in the target database for the following rows:

- extents used
- io errors
- page gaps
- page extent crosses
- page extent gaps
- page format errors
- pages reserved
- pages overhead
- pages misallocated
- pages not allocated
- pages not referenced
- pages used

The page gaps, page extent crosses, and page extent gaps indicate how the data pages for the objects are distributed on the database devices. Large values indicate less effectiveness in using larger buffer sizes and in data prefetch.

- If multiple dbcc checkstorage operations were run on a target database on the same day, sp_dbcc_statisticsreport generates a report based on the results of the last dbcc checkstorage operation that finished before the specified time.

Permissions

Any valid user for the database name specified can run sp_dbcc_statisticsreport.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_fullreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_summaryreport

Description

Generates a summary report on the specified database.

Syntax

sp_dbcc_summaryreport [*dbname* [, *date*] [, *opname*]]

Parameters

dbname

specifies the name of the database for which you want the report generated. If you do not specify *dbname*, sp_dbcc_summaryreport generates reports on all databases in dbccdb..dbcc_operation_log for which the date is on or before the date and time specified by the *date* option.

date

specifies the date on which dbcc checkstorage was performed. If you do not specify a date, sp_dbcc_summaryreport uses the date of last dbcc checkstorage operation performed on the **target database**. This parameter is of the datatype datetime. If both the date and the time are specified for *date*, summary results of all the operations performed on or before the specified time are reported. If no date is specified, all operations are reported.

opname

specifies the operation. *opname* may be either checkstorage, which is the default, or checkverify, or both. If *opname* is not specified, reports are generated for all operations.

Examples

Example 1 Generates a summary report on the sybsystemprocs database, providing information on all dbcc checkstorage and dbcc checkverify operations performed:

```
sp_dbcc_summaryreport

DBCC Operation : checkstorage

Database Name      Start time           End Time    Operation ID
    Hard Faults Soft Faults Text Columns Abort Count
       User Name
------------------ -------------------- ----------- ------------
    ----------- ----------- ------------ -----------
       --------------------------
sybsystemprocs     05/11/1999 14:53:11  14:53:32:163            1
              0           0            0           0
       sa
sybsystemprocs     05/11/1999 14:55:06  14:55:29:200            2
              0           0            0           0
       sa
sybsystemprocs    05/11/1999 14:56:10  14:56:27:750             3
              0           0            0           0
```

```
        sa

DBCC Operation : checkverify

Database Name      Start time           End Time    Operation ID
    Hard Faults    Soft Faults     User Name
------------------ -------------------- ----------- ------------
    -------------- --------------- ---------------------
sybsystemprocs     05/11/1999 14:55:29  14:55:29:310            2
                 0               0 sa
```

Example 2 Generates a summary report on the user database testdb, providing information on all dbcc checkstorage operations performed. dbcc checkstorage was the only operation run on this database, so no dbcc checkverify information appears on the report:

```
sp_dbcc_summaryreport "testdb"
```

```
DBCC Operation : checkstorage

Database Name   Start time           End Time    Operation ID
    Hard Faults Soft Faults Text Columns Abort Count  User Name
--------------- -------------------- ----------- ------------
    ----------- ----------- ------------ -----------  ----------------
testdb         05/11/1999 14:55:29  14:55:49:903  1
     0           0            0           0           sa
 testdb         05/11/1999 14:55:50  14:56:9:546   2
     0           0            0           0           sa
 testdb         05/11/1999 14:56:28  14:56:40:666  3
     0           0            0           0           sa
```

Example 3 Generates a summary report on the sybsystemprocs database, providing information on all dbcc checkverify operations performed. Because dbcc checkverify was the specified operation, no dbcc checkstorage information appears on the report:

```
sp_dbcc_summaryreport null, null, "checkverify"
```

```
DBCC Operation : checkverify
Database Name   Start time           End Time    Operation ID
    Hard Faults    Soft Faults     User Name
--------------- -------------------- ----------- ------------
    -------------- --------------- ---------------------
sybsystemprocs    05/11/1999 14:55:29   14:55:29:310          2
     0               0              sa
```

Example 4 Generates a summary report on the sybsystemprocs database, providing information on all dbcc checkstorage operations performed. Because dbcc checkstorage was the specified operation, no dbcc checkverify information appears on the report:

```
sp_dbcc_summaryreport sybsystemprocs, null, "checkstorage"
```

```
DBCC Operation : checkstorage

Database Name   Start time           End Time    Operation ID
    Hard Faults Soft Faults Text Columns Abort Count  User Name
--------------- -------------------- ------------ ------------
    ----------- ----------- ------------ -----------  ---------
sybsystemprocs   05/11/1999 14:53:11  14:53:32:163  1
    0           0            0           0            sa
sybsystemprocs   05/11/1999 14:55:06  14:55:29:200  2
    0           0            0           0            sa
sybsystemprocs   05/11/1999 14:56:10  14:56:27:750  3
    0           0            0           0            sa
```

Usage

- sp_dbcc_summaryreport generates a summary report of checkstorage or checkverify operations, or both, on the specified database.
- The report indicates the name of the database that was checked, the start and end time of the dbcc checkstorage run and the number of soft and hard faults found.
- The “Operation ID” column contains a number that identifies the results of each dbcc checkstorage operation on a given database at a specific time. The number provided in the report comes from the opid column of the dbcc_operation_log table. See the *System Administration Guide* for more information.
- The “Text Columns” column shows the number of non-null text columns found by dbcc checkstorage during the run.
- The “Abort Count” column shows the number of tables that contained errors, which caused dbcc checkstorage to abort the check on the table. For details on the errors, run sp_dbcc_faultreport.

Permissions

Any valid user for the database name specified can run sp_dbcc_summaryreport.

See also

Commands dbcc

dbcc stored procedures sp_dbcc_fullreport, sp_dbcc_statisticsreport, sp_dbcc_updateconfig

sp_dbcc_updateconfig

Description

Updates the dbcc_config table in dbccdb with the configuration information of the target database.

Syntax

sp_dbcc_updateconfig *dbname*, *type*, "*str1*" [, "*str2*"]

Parameters

dbname

is the name of the target database for which configuration information is being updated.

type

specifies the type name from the dbcc_types table. Table 4-2 on page 544 shows the valid values for *type*.

str1

specifies the first configuration value for the specified *type* to be updated in the dbcc_config table. Table 4-2 on page 544 describes the expected value of *str1* for the specified *type*.

str2

specifies the second configuration value for the specified *type* that you want to update in the dbcc_config table. Table 4-2 on page 544 describes the expected value of *str2* for the specified *type*.

Examples

Example 1 Updates dbcc_config with the maximum number of worker processes for dbcc checkstorage to use when checking the pubs2 database. The new maximum number of worker processes is 4:

```
sp_dbcc_updateconfig pubs2, "max worker processes", "4"
```

Example 2 Updates dbcc_config with the size of the dbcc named cache “pubs2_cache”. The new size is 10K:

```
sp_dbcc_updateconfig pubs2, "dbcc named cache", pubs2_cache, "10K"
```

Example 3 Updates dbcc_config with the new name of the scan workspace for the pubs2 database. The new name is scan_pubs2. This update is made after using sp_dbcc_alterws to change the name of the scan workspace:

```
sp_dbcc_updateconfig pubs2, "scan workspace", scan_pubs2
```

Example 4 Updates dbcc_config with the new name of the text workspace for the pubs2 database. The new name is text_pubs2. This update is made after using sp_dbcc_alterws to change the name of the text workspace:

```
sp_dbcc_updateconfig pubs2, "text workspace", text_pubs2
```

Example 5 Updates dbcc_config with the OAM count threshold value for the pubs2 database. The new value is 5:

```
sp_dbcc_updateconfig pubs2, "OAM count threshold", 5
```

Example 6 Updates dbcc_config with the I/O error abort value for the pubs2 database. The new value is 3:

```
sp_dbcc_updateconfig pubs2, "IO error abort", 3
```

Example 7 Updates dbcc_config with the linkage error abort value for the pubs2 database. The new value is 8:

```
sp_dbcc_updateconfig pubs2, "linkage error abort", 8
```

Usage

- sp_dbcc_updateconfig updates the dbcc_config table for the target database.
- If the name of the target database is dbccdb, and the database dbccalt exists, sp_dbcc_updateconfig updates the dbcc_config table in dbccalt.
- If the target database name is not found in dbcc_config, sp_dbcc_updateconfig adds it and sets the operation sequence number to 0 before updating other configuration information.
- If the expected value for the specified *type* is a number, sp_dbcc_updateconfig converts the values you provide for *str1* and *str2* to numbers.
- Table 4-2 shows the valid type names to use for *type* and the expected value for *str1* or *str2*.

Table 4-2: Type names and expected values

***type* name**	**Value expected for *str1* or *str2***
dbcc named cache	The name of the cache, specified by *str1*, and the new size (in kilobytes or megabytes) or the number of 2K pages, specified by *str2*.
IO error abort	The new error count, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
linkage error abort	The new linkage error count value specified in *str1*. The value must be a number greater than 0. *str2* is not used with this type.
max worker processes	The new number of worker processes, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
OAM count threshold	The new threshold count, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
scan workspace	The new name for the scan workspace, specified by *str1*. *str2* is not used with this type.
text workspace	The new name of the text workspace, specified by *str1*. *str2* is not used with this type.

- Se the *System Administration Guide* for more information on the *type* names and values.

Permissions	Only a System Administrator or the Database Owner can run sp_dbcc_updateconfig.
See also	**Commands** dbcc
	dbcc stored procedures sp_dbcc_alterws, sp_dbcc_evaluatedb
	System procedures sp_plan_dbccdb

Index

Symbols

@ (at sign) and procedure parameters 3
::= (BNF notation)
  in SQL statements xvi
, (comma)
  in SQL statements xvii
  in user-defined datatypes 61
{} (curly braces)
  in SQL statements xvi
.. (dots) in database object names 37
() (parentheses)
  in SQL statements xvi
  in user-defined datatypes 61
“ ” (quotation marks)
  enclosing parameter values 3, 474
  enclosing reserved words 109
  single, and **quoted_identifier** 115
[ ] (square brackets)
  in SQL statements xvii

Numerics

0 return status 1, 474
7-bit ASCII characters, checking with **sp_checknames** 103
7-bit terminal, **sp_helpsort** output 312
8-bit terminal, **sp_helpsort** output 312

A

abort tran on log full database option 160
abstract plan groups
  adding 40
  dropping 207
  exporting 234
  importing 320
  renaming 402
abstract plans
  information about 299
  viewing with **sp_help_qplan** 299
accounting, chargeback
  sp_clearstats 124
  sp_reportstats 403–404
accounts. *See* logins.
actions
  modifying for resource limits 350
  resource limit information on 302
  specifying for resource limits 45
adding
  abstract plan groups 40
  aliases 16–39
  date strings 29–31
  dump devices 65–66
  engine groups 21
  engines to a group 21
  execution classes 22
  foreign keys 250–251
  group to a database 28
  limits 44
  logins to Server 32–34
  messages to *sysusermessages* 35–36
  named time ranges 58
  remote logins 41–43
  resource limits 44
  segments 49–50
  servers 51–53
  thresholds 54–57
  time ranges 58
  user-defined datatypes 61–64
  users to a database 67–68
  users to a group 67–68, 102
adhoc auditing option 70
aliases, language
  assigning 413
  defining 29–31
aliases, server 52
aliases, user

See also logins; users
assigning 16
assigning different names compared to 67
database ownership transfer and 101
dropping 191, 220
help on 317
sysalternates table 16, 191
all auditing option 70
allow nulls by default database option 160
ALS (Asynchronous Log Service) 278
ALS (Asynchronous Log Service) output in sp_helpdb 279
alter auditing option 70
alter database command
sp_dbremap and 167
alternate identity. *See* aliases, user
alternate languages. *See* languages, alternate
ANYENGINE engine group 21
applications
applying resource limits to 44
dropping resource limits from 210
modifying resource limits for 349
resource limit information on 301
ASCII characters
checking for with **sp_checknames** 103
Asynchronous Log Service (ALS) 278
output in sp_helpdb 279
asynchronous prefetch
configuring limits 386
at sign (@)
procedure parameters and 3
@@connections global variable and **sp_monitor** and 362
@@cpu_busy global variable and **sp_monitor** 362
@@idle global variable
sp_monitor and 362
@@io_busy global variable
sp_monitor and 362
@@ncharsize global variable
sp_addtype and 63
@@pack_received global variable
sp_monitor and 362
@@pack_sent global variable and **sp_monitor** 362
@@packet_errors global variable and **sp_monitor** and 362
@@thresh_hysteresis global variable
threshold placement and 55
@@total_errors global variable
sp_monitor and 362
@@total_read global variable
sp_monitor and 362
@@total_write global variable
sp_monitor and 362
attributes
execution classes 22
server (**sp_server_info**) 486
sp_addobjectdef and 39
audit trail, adding comments 17
auditing
adding an audit table 19
options, displaying 181
auditing options
adhoc 70
all 70
alter 70
bcp 70
bind 70
cmdtext 70
create 70
dbaccess 70
dbcc 70
delete 70
disk 70
drop 70
dump 70
errors 70
exec_procedure 70
exec_trigger 70
func_dbaccess 70
func_obj_access 70
grant 70
insert 70
load 70
login 70
logout 70
reference 71
revoke 71
rpc 71
security 72
select 73
setting 70
setuser 73
table_access 73

truncate 73
unbind 73
update 73
view_access 73
authority. *See* permissions.
authorizations. *See* permissions.
auto identity database option 160

B

Backup Server
See also Utility Guide
amount dumped, specifiying 221
information about 311
multiple 53
volume handling messages 466–468
Backus Naur Form (BNF) notation xvi
basic display level for configuration parameters 185
bcp (bulk copy utility)
select into/bulkcopy/pllsort and 163
bcp auditing option 70
binary sort order of character sets 313
bind auditing option 70
binding
data caches 78–81
defaults 82–83
objects to data caches 78–81
rules 88–89
unbinding and 457–458, 460
user messages to constraints 87
blanks
catalog stored procedure parameter values 474
in system procedure parameter values 3
blocking process
sp_lock report on 238, 333
sp_who report on 471
BNF notation in SQL statements xvi
brackets. *See* square brackets [ ]
bulk copying. *See* **bcp** (bulk copy utility)

C

caches, data
binding objects to 78
configuring 90–97
dropping 97
information about 93, 267
logonly type 96
memory pools 382–386
overhead 96, 267
recovery and 93
status 95
unbinding all objects from 459
unbinding objects from 457
case sensitivity in SQL xviii
catalog stored procedures 473–497
list of 474
return status 474
syntax 474–475
chained transaction mode
sp_procxmode and 391
changing
database options 157–164
database owners 101
dbccdb workspace size 520
language alias 413
memory pools within data caches 382
names of abstract plan groups 402
object names 398–399
passwords for login accounts 376–377
resource limits 349
thresholds 357–360
time ranges 352
user's group 102
character sets
changing names of 112, 114
checking with **sp_checknames** 103
checking with **sp_checkreswords** 108
multibyte 313
sp_helpsort display of 312
chargeback accounting
sp_clearstats procedure 124–125
sp_reportstats procedure 403–404
check constraints
binding user messages to 87
displaying source text of 314
renaming 398–399
checking passwords. *See* passwords; **sp_remoteoption** system procedure.
checkpoint command

setting database options and 159
clearing accounting statistics 124–125
clustered indexes, *indid* not equal to one 287
cmd returned by **sp_who** 471
cmdtext auditing option 70
codes
datatype 482
ODBC datatype 476
collating sequence. *See* sort order
collisions, hash key 298
column data. *See* datatypes
column name
changing 111, 398–399
checking with **sp_checknames** 103
column pairs. *See* joins; keys
columns
common key 132–133
datatypes 479
defaults for 82–83
dependencies, finding 111
foreign keys 250–251, 483
joins and 290
permissions on 477
primary key 387
returned by **sp_who** 471
rules 88–89
unbinding defaults from 460
unbinding rules with **sp_unbindrule** 464–465
comma (,)
in SQL statements xvii
in user-defined datatypes 61
commands, display syntax of 435–436
comments, adding to audit trail 17
common keys
See also foreign keys; joins; primary keys
defining 132–133
dropping 200
join candidates and 290
reporting 292–293
companion servers
configuring 134–136
comparing plan groups 128
comparing plans 128, 130
compiled objects
checking for source text of 117
displaying source text of 314
hiding source text of 318
compiling and **sp_recompile** 392
comprehensive display level for configuration parameters 185
concurrency optimization 120
concurrency_opt_threshold option, **sp_chgattribute** 120
configuration parameters
changing 137–142
display levels 185
help information on 269
constraints
binding user messages to 87
displaying source text of 314
information about 264, 274
renaming 398–399
unbinding messages with **sp_unbindmsg** 463
contention, lock
monitoring with **sp_object_stats** 371–373
controller, device
sp_helpdevice and number 281
conventions
See also syntax
Transact-SQL syntax xvi
used in the Reference Manual xvi
copying
plan groups 143
plans 143, 144
corrupt databases
listing 329
recovery fault isolation mode 420
corrupt pages
bringing online 248–249
isolating on recovery 420–422, 423
listing 331
CPU usage, monitoring 362
@@cpu_busy global variable and **sp_monitor** 362
create auditing option 70
create database command
log on option compared to **sp_logdevice** 338
create index command
sp_extendsegment and 235
create table command
sp_extendsegment and 235
creating
abstract plan groups 40

datatypes 61–64
dbccdb workspaces 522
execution classes 22
extended stored procedures 23–24
limits 44
named time ranges 58
resource limits 44
thresholds 54–57
time ranges 58
user aliases 16
user groups 28
user-defined audit records 70
curly braces ({}) in SQL statements xvi
current database
information from **sp_helpdb** 280
space used by 430–432
current locks, **sp_lock** system procedure 332
current usage statistics 403–404
cursors 147
custom audit records 70
custom datatypes. *See* user-defined datatypes

D

data caches
binding objects to 78
configuring 90–97
dropping 97
information about 93, 267
logonly type 96
memory pools 382–386
overhead 96, 267
recovery and 93
status 95
unbinding all objects from 459
unbinding objects from 457
data dependency. *See* dependencies, database object
database design
dropping keys 200
logical relationships in 132, 250
database devices
defaulton or **defaultoff** status 179–180
dropping 193
dropping segments from 214–215
dsynch setting of 177
listing of 281
sp_helpdevice system procedure 281
status 179
database files. *See* files.
database object owners
sp_depends system procedure and 171
database objects
binding defaults to 82–83
binding rules to 88
binding to caches 78
dependencies of 171–176
finding 175, 261
listings of 257
permissions on 304
remapping 393
renaming 398–399
sp_tables list of 497
space used by 430–432
database options 160–164
See also individual option names
listing 157–164
showing settings 159, 278
database owners
See also database object owners; permissions
adding users 67
changing 101
dbo use only database option 161
information about 317
transferring ownership 101
database recovery order
sp_dbrecovery_order system procedure 165–166
system databases and 166
databases
See also database objects
adding groups 28
adding users 67
binding to data caches 78, 79
changing user's default 346
checking with **sp_checknames** 103
dropping row lock promotion thresholds for 213
dropping segments from 214–215
dropping users from 220
help on 278
information on storage space used 280, 430
listing suspect 329
listing suspect pages in 331

listing with **sp_databases** 481
listing with **sp_helpdb** 278
lock promotion thresholds for 415
options 157–164
ownership 101
renaming 400–401
running out of space in 448
setting row lock promotion thresholds for 418
storage information 430
thresholds 448
unbinding from data caches 457

datatype precedence. *See* precedence

datatypes
codes 476, 482
defaults and 82–83
dropping user-defined 219
hierarchy 63
ODBC 476
physical 61
sp_datatype_info information on 482
sp_help information on 257–263
unbinding defaults from 460
unbinding rules with **sp_unbindrule** 464–465

datatypes, custom. *See* user-defined datatypes

date parts, order of 29

day-long time ranges 58

days
alternate language 29
in time ranges 58

dbaccess auditing option 70

dbcc (Database Consistency Checker)
scripts and **sp_checkreswords** 110
space allocation and 378

dbcc auditing option 70

dbccdb database
changing workspace size in 520
creating workspaces in 522
deleting **dbcc checkstorage** history from 525
deleting target database information from 524
reporting allocation statistics from 537
reporting comprehensive information from 534
reporting configuration information from 521, 531, 534
reporting fault information from 527, 531
reporting full details from 534
reporting I/O statistics from 527
stored procedures for use with 517

DB-Library programs and changing identifier names 110

dbo use only database option
setting with **sp_dboption** 160

ddl in tran database option 161

default database
See also sysdevices table
assigning with **sp_addlogin** 32
changing user's 346

default database devices
setting status with **sp_diskdefault** 179
sp_helpdevice and 281

default language id configuration parameter 32

default segment
dropping 214
mapping 50

default settings
changing login 34, 346
configuration parameters 140
language 32

defaulton | defaultoff option, **sp_diskdefault** 179

defaults
binding 82–83
checking name with **sp_checkreswords** 107
displaying source text of 314
remapping 393
renaming 110, 398–399
system tables and 83
unbinding 460

defncopy utility command 109

delete auditing option 70

deleting
See also dropping
dbcc checkstorage history from *dbccdb* 525
files 193
plans 192, 208
target database information from *dbccdb* 524

delimited identifiers
testing 109
using 108, 115

denying access to a user 336

dependencies, database object
changing names of 109
recompilation and 399
sp_depends system procedure 171–176

detail option, **sp_helpconstraint** 274
device fragments, **sp_helpdb** report on 278
devices
  See also sysdevices table
  changing names of 112, 114
  dsync setting for 177
  information on log 295
direct updates
  to system tables 112
disk auditing option 70
disk devices
  adding 65–66
disk mirroring
  sp_who report on 472
disk option, **sp_addumpdevice** 65
display
  character sets 312
  database options 157–164
  source text of compiled objects 314
  syntax of modules 435
distributed Transaction Management (DTM) 450
dots (..) for omitted name elements 37
drop auditing option 70
drop logins option, **sp_dropserver** 216
dropmessages option, **sp_droplanguage** 202
dropping
  See also deleting
  abstract plan groups 207
  aliased user 191
  database devices 193
  groups 199
  lock promotion thresholds 198
  plans 192, 208
  procedures 196
  remote logins 209, 216
  remote servers 216
  resource limits 210
  row lock promotion thresholds 213
  segment from a database 214–215
  time ranges 218
  user from a database 220
  user from a group 102
  user-defined datatype 219
  user-defined messages 204
  workspaces 523
dsync setting 177
dump auditing option 70
dump devices
  See also database devices; log device
  adding 65–66
  dropping 193
  listing 281
  permission and ownership problems 66
dump transaction command and **sp_logdevice** 338
dump, database
  interrupted 167
dumping databases 221
dumpvolume option
  dump database 466
Dynamic Link Libraries (DLLs), unloading 252

E

8-bit terminal, **sp_helpsort** output 312
encryption
  compiled object source text 318
  reversing 318
ending days of named time ranges 58
ending times of named time ranges 58
enforcing resource limits 45
errors
  number of 362
errors auditing option 70
exclusive locks 237, 333
exclusive row locks 335
exec_procedure auditing option 70
exec_trigger auditing option 70
execute to public, sp_dbextend 156
executing operating system commands 501
exp_row_size option
  sp_chgattribute 119
  sp_help report on 262
expand_down parameter
  sp_activeroles 15
  sp_displayroles 189
  sp-displayroles 189
exporting plan groups 234
extended stored procedures
  creating 23–24
  displaying 283
  dropping 196

extending
  segments 235
external engine auto start option
  sp_serveroption 409

F

failures, media and **trunc log on chkpt** database option 164
family of worker processes
  fid reported by **sp_lock** 334
  sp_familylock report on *fid* 237
fault isolation
  index level 246, 330
fid (family ID) number 237
  sp_lock report 334
file names
  configuration file 138
  DLL 252
files
  See also tables; transaction log
  deleting 193
  inaccessible after **sp_dropdevice** 193
  interfaces, and server names 52
  localization 114
finding
  cache bindings 90, 267
  character sets 312
  configuration parameters 269
  constraints 274
  database objects 261
  database options 157
  database settings 278
  datatypes 257
  devices 281
  languages 294
  object dependencies 171, 175
  object information 257
  partition information 264
  permissions 304
  reserved words 104
  resource limits 301
  segments 308
  server names 311
  thresholds 316
  users in a database 317
first page
  log device 295
  partition, displaying with **sp_helppartition** 264
foreign keys
  dropping 200
  inserting 250–251
  sp_fkeys information on 483
  sp_helpkey and 292
format strings in user-defined error messages 36
formats, times in named time ranges 58
formulas for **max_rows_per_page** of nonclustered indexes 122
fragments, device space and **sp_placeobject** 378
from keyword
  sp_tables list of objects appearing in clause 497
full name
  changing with **sp_modifylogin** 346
  specifying with **sp_addlogin** 34
func_dbaccess auditing option 70
func_obj_access auditing option 70
future space allocation. *See* space allocation; **sp_placeobject** system procedure
futureonly option
  sp_bindefault 82
  sp_bindrule 88, 89
  sp_unbindefault 460
  sp_unbindrule 464

G

getting messages. *See* **sp_getmessage** system procedure.
global variables
  See also individual variable names
  sp_monitor report on 361
grant auditing option 70
grant option
  sp_helprotect 304
  sp_role 406
groups
  See also “public” group
  changing 102
  dropping 199
  information about 285

  sp_addgroup 28
  sp_adduser procedure 67
  Windows NT domain 505
guest users and *sybsystemprocs* database 2

H

hash-key collisions 298
help
  sp_syntax display 435
  sp_sysmon display 437
help reports
  See also information (server); system procedures
  constraints 274
  database devices 281
  database object 257
  databases 278
  datatypes 257
  dump devices 281
  extended stored procedures 283
  groups 285
  indexes 286
  joins 290
  keys 292
  language, alternate 294
  logins 300
  permissions 304
  remote servers 311
  resource limits 301
  segments 308
  source text for compiled objects 314
  system procedures 257–317
  tables 257
  thresholds 316
  users 317
hierarchy
  See also precedence
  data cache bindings 79
  lock promotion thresholds 415, 419
  roles, displaying with **sp_activeroles** 15
  user-defined datatypes 63
hierarchy of roles. *See* role hierarchies
high availability
  configuring Adaptive Server for 134
holdlock keyword
  select 333

I

I/O
  concurrency_opt_threshold and 120
  configuring size 382
  limiting 45
  log size 343
  usage statistics 403
identifiers
  delimited 108
  quoted 108
  renaming 109
  reserved words and 104–116
  set quoted_identifier on 108, 115
  sp_checkreswords and 108
identities
  alternate 16
IDENTITY columns
  automatic 160, 164
  database options using 162
  nonunique indexes 162
identity in nonunique index database option, setting with **sp_dboption** 162
identity keyword
  sp_addtype and 61
identity of user. *See* aliases; logins; users
identity_gap option
  sp_chgattribute 120
@@*idle* global variable
  sp_monitor and 362
IDs, time range 59
IDs, user
  See also logins
image datatype, size of 430
importing abstract plan groups 320
index pages
  locks on 334
indexes
  binding to data caches 78
  checking name with **sp_checknames** 103
  checking name with **sp_checkreswords** 107
  estimating space and time requirements 229
  IDENTITY columns in nonunique 162

information about 286
order of, reported by **sp_helpindex** 287
renaming 110, 398–399
sp_placeobject space allocation for 378–379
sp_statistics information on 493
space used by 431
suspect 322
unbinding from data caches 457

information (server)
current locks 332
indexes 286
suspect indexes 322

information about, reporting
cache bindings 80
current locks 332
data caches 93
database devices 281
database objects 257
database owners 317
databases 278
datatypes 257
dump devices 281
extended stored procedures 283
first page of log 295
groups 285, 317
indexes 286
join columns 290
keys 292
languages 294
locks 332, 371
logins 469
performance 437
permissions 304
remote server logins 300
remote servers 311
resource limits 301
segments 308
server processes 469
server users 186, 469
source text for compiled objects 314
space usage 430
statistics, monitor 361
suspect indexes 322
thresholds 316
transaction log device 295
users, database 317

input packets, number of 362
insert auditing option 70
installdbextend script, permissions 156
intent table locks 237, 333
interfaces file
changing server names in 114
sp_addserver and 52
intermediate display level for configuration parameters 185
@@*io_busy* global variable
sp_monitor and 362
isolation levels
catalog stored procedures 474
identity in nonunique index database option and 162
system procedures 1

J

Java items and **sp_helpjava** system procedure 288
joins
information about 290
sp_commonkey 132

K

keys, table
See also common keys; indexes
dropping 200
information about 292
syskeys table 132, 250, 387
keywords
as identifiers 104
kill command and **sp_who** 471

L

language defaults 32
adding 29–31
changing user's 34
languages, alternate
alias for 413
changing names of 112, 114

checking with **sp_checkreswords** 108
date formats in 29
dropping 202
dropping messages in 204
information on 294
installing 29
official name 413
syslanguages table 294
system messages and 253
user-defined messages 35
without Language Modules 29
last-chance thresholds 55, 358
LASTONLINE engine group 21
LDAP, URL search strings 323
limit types 44
elapsed time 44
I/O cost 44
modifying values 349
number of rows returned 44
specifying values 44
tempdb_space 44
limited days
modifying for time ranges 352
resource limit information on 301
specifying for time ranges 58
limited times
modifying for time ranges 352
resource limit information on 301
specifying for time ranges 58
linking users. *See* alias, user
listing
database options 157
devices 281
lists
catalog stored procedures 473
dbcc stored procedures 517
system procedures 1–14
load auditing option 70
local alias, language 413
local option, **sp_addserver** 51
local servers 51
See also remote servers; servers
localization, changing language names and files 114
lock | unlock option, **sp_locklogin** 336
lock promotion thresholds 414
dropping row with **sp_droprowlockpromote** 213
setting row with **sp_setrowlockpromote** 418
sp_help report on 262
sp_helpdb report on database setting 278
locking
cache binding and 80
cache unbinding and 458
control over 332–335
logins 336
monitoring contention 371
locking scheme, **sp_help** report on 262
locks
displaying information about 332, 371
exclusive page 237
exclusive table 237
exclusive table and page 333
“FAM DUR” status 238
intent table 237, 333
page 237, 333
reported by **sp_lock** 332
row 334
shared page 237, 333
shared table 237, 333
sp_familylock system procedure 237–239
sp_lock system procedure 332–335
types of 237, 333
log device information 295
log on option
create database, and **sp_logdevice** 338
log segment
sp_helplog report on 295
sp_helpthreshold report on 316
logging
messages 69
user-defined events 507
user-defined messages 35
logical device name 65, 179
syslogs table 338
login auditing option 70
logins
See also remote logins; users
accounting statistics 124, 403
adding to Servers 32–34
alias 16, 191
applying resource limits to 44
changing current database owner 101

dropping 203, 216
dropping resource limits from 210
information about 300
information on 186
locking 336–337
modifying accounts 346–348
modifying resource limits for 349
number of 362
options for remote 394
password change 376–377
“probe” 404
remote 209, 216
resource limit information on 301
sysremotelogins table 41–43, 209, 216
unlocking 336–337
logout auditing option 70
logsegment log storage, dropping 214
lower and higher roles. *See* role hierarchies

M

machine ticks 362
mail messages, server
deleting 504
processing 388–389
reading 508
sending 511
starting session 515
stopping session 516
mapping
databases 167
remote users 41
master database
checking with **sp_checkreswords** 107
sp_dboption and 159
system procedure tables 5
thresholds and 56, 359
max_rows_per_page option
changing with **sp_relimit** 119
sp_chgattribute 119
memory
freeing from XP Server 252
mapping 167
used by configuration parameters 269
memory pools
configuring 382
configuring asynchronous prefetch limits 386
configuring wash percentage 385
defaults 91
minimum size of 384
sp_logiosize and 343
transaction logs and 385
message **output** parameter, **sp_getmessage** 253
messages
adding user-defined 35–36
dropping system with **sp_droplanguage** 202
dropping user-defined 204
language setting for 204, 253
logging 69
number for 35, 69, 204, 253
sp_getmessage procedure 253
system procedure 4
sysusermessages table 35–36
unbinding with **sp_unbindmsg** 463
mirroring. *See* disk mirroring
model database, changing database options 159
modifying
configuration parameter display level 185
configuration parameters 137
login accounts 346
named time ranges 352
resource limits 349
thresholds 357
modifying abstract plans 417
modules, display syntax of 435
monitoring
lock contention 371
space remaining 54, 55, 358
system activity 361
month values
alternate language 29
moving
indexes 378
tables 378
transaction logs 338
user to new group 102
MRU replacement strategy
disabling 98
multibyte character sets
sort order 313
sp_helpsort output 313

mutual authentication option, **sp_serveroption** 409

N

named time ranges
- adding 58
- “at all times” 59, 218
- changing active time ranges 60
- creating 58
- dropping 218
- entire day 58
- IDs for 59
- modifying 352
- overlapping 59

names
- alias 16, 191, 220
- assigning different, compared to aliases 67
- changing database object 398–399
- changing identifier 109
- checking with **sp_checknames** 103
- checking with **sp_checkreswords** 104
- DLL file 252
- remote user 209
- server 51
- server attribute 486
- user’s full 32

naming
- groups 28
- time ranges 58
- user-defined datatypes 63

@@*ncharsize* global variable
- **sp_addtype** and 63

nesting cursors 147

net password encryption option
- **sp_serveroption** 409

no chkpt on recovery database option, setting with **sp_dboption** 162

no free space acctg database option, setting with **sp_dboption** 162

not null values
- **sp_addtype** and 62
- for user-defined data 62

null values
- **sp_addtype** and 61
- for user-defined datatypes 61

number (quantity of)
- databases reported by **sp_countmetadada** 145
- groups per user 102
- indexes 145
- messages per constraint 87
- open objects 145

numbers
- *See also* IDs, user
- datatype code 476
- device 281
- global variable unit 362
- message 35, 69, 204, 253
- ODBC datatype code 476
- *spid* (server process ID) 469
- weekday names and 29

O

object names, database
- checking with **sp_checknames** 103
- checking with **sp_checkreswords** 107

object owners. *See* database object owners.

objects. *See* database objects; databases.

ODBC. *See* Open Database Connectivity (ODBC) API datatypes.

official language name 30, 413
- *See also* aliases; languages, alternate

Open Client applications
- connection security with 42

Open Database Connectivity (ODBC) API datatypes 476

operating system commands 501

optdiag utility
- flushing in-memory statistics 244

optimistic index locking 119, 120, 121, 257, 261

optimization
- queries (**sp_recompile**) 392

options
- *See also* configuration parameters
- database 157–164
- remote logins 394–395
- remote servers 409–412

order
- *See also* indexes; precedence; sort order
- of date parts 29

output option
  sp_getmessage 253
output, number of
  packets 362
overhead, data caches 96
overlapping time ranges 59
owners. *See* database object owners; database owners
ownership and dump devices 66

P

@@pack_received global variable
  sp_monitor and 362
@@packet_errors global variable and **sp_monitor** and 362
page locks, types of 237, 333
pages, data
  computing number of, with **sp_spaceused** 431
  locks held on 237, 333
pair of columns. *See* common keys; joins
parameters, procedure
  ways to supply 3, 474
parentheses ()
  in SQL statements xvi
  in user-defined datatypes 61
passthrough mode
  sp_autoconnect system procedure 76
  sp_passthru system procedure 374
  sp_remotesql system procedure 396
passwords
  date of last change 187
  encryption over network 411
  setting with **sp_addlogin** 32
  sp_password 376–377
  sp_remoteoption and 394
  sp_serveroption and 411
  trusted logins or verifying 394
path name, dump device 65
pattern matching and catalog stored procedure parameters 475
PC DB-Library. *See* DB-Library programs
performance
  concurrency optimization 120
  information about 437
permissions
  displaying user's 186
  dump devices and 66
  granting 304
  information on 304
  new database owner 101
  new database user 348
  revoking 304
  sp_column_privileges information on 477
  system procedures 2
physical datatypes 61
physical device name 65
placeholders
  error message percent sign (%) 36
plan groups
  adding 40
  comparing 128
  copying 143
  copying to a table 234
  creating 40
  dropping 207
  dropping all plans in 192
  exporting 234
  information about 297
  reports 297
plans
  changing 417
  comparing 128, 130
  copying 143, 144
  deleting 192
  dropping 192, 208
  finding 240
  modifying 417
  searching for 240
  sp_showplan output 427
pools, memory
  configuring 382
  defaults 91
precedence
  binding defaults to columns and datatypes 83
  resource limits 47
  rule binding 89
precision, datatype
  sp_help report on 260
  user-defined datatypes 61
prefetch
  disabling 98

enabling 98
primary keys
  sp_dropkey procedure 200
  sp_foreignkey and 250
  sp_helpkey and 292
  sp_primarykey definition of 387
priority
  sp_setpsexe 416
“probe” login account 404
probe process, two-phase commit 404
procedures. *See* stored procedures; system procedures
process logical name. *See* logical device name
processes (server tasks)
  checking locks held 332
  checking locks on 237–239, 332–335
  ID number 469
  sp_showplan display of 427–428
  sp_who report on 469–472
promotion, lock 414
protection system
  groups 28
  locking logins 336
“public” group
  See also groups
  information report 285
  sp_addgroup and 28
  sp_adduser and 67
  sp_changegroup and 102
  sp_helpgroup report on 285
punctuation
  enclosing in quotation marks 3, 474
  in user-defined datatypes 61

Q

queries
  compilation and optimization 392
  sp_tables and 497
query plans, recompiling with **sp_recompile** 392
query processing, limiting with
    sp_add_resource_limit 44
quotation marks (“ ”)
  enclosing parameter values 3, 474
  enclosing reserved words 109
  single, and **quoted_identifier** 115
quoted identifiers
  testing 109
  using 108, 115

R

range locks 334
range, specifying for resource limits 44
read only database option
  setting with **sp_dboption** 163
  setting with **sp_setsuspect_granularity** 420
readonly option, **sp_serveroption** 409
recompilation
  stored procedures 392
records, audit 17
recovery
  data caches and 93
  displaying mode 420
  forcing suspect pages online with
    sp_forceonline_db 245
  forcing suspect pages online with
    sp_forceonline_page 248
  listing offline pages 331
  listing suspect databases 329
  setting mode 420
  setting threshold 423
recovery fault isolation 246, 330
reference auditing option 71
reference information
  catalog stored procedures 473
  dbcc stored procedures 517
  system extended stored procedures 499
  system procedures 1–5
referencing, object. *See* dependencies, database object
referential integrity constraints
  binding user messages to 87
  renaming 398–399
regulations
  for finding objects 175, 261
reindex option, **dbcc**
  after **sp_indsuspect** 322
remapping database objects 393
remote logins
  See also logins; users
  dropping 209

information on 300
sp_remoteoption for 394–395
sysremotelogins table 41–43
trusted or untrusted mode 394
remote procedure calls
sp_password 377
remote servers
See also servers
changing names of 112, 114
dropping logins 209
information on 311
information on logins of 300
names of 51
passwords on 377
sp_remoteoption and 394–395
remote users. *See* remote logins
removing. *See* dropping; deleting
renaming 398–399
See also **sp_rename** system procedure
a database 400–401
warnings about 399, 401
replacing user-defined messages 35
reporting from *dbccdb* database
allocation statistics 537
comprehensive information 534
configuration information 521, 531, 534
fault information 527, 531
full details 534
I/O statistics 527
reports
plan groups 297
sp_who 469–472
reserved words
catalog stored procedures and 474
as identifiers 104–116
system procedures and 3
reservepagegap option
sp_chgattribute 119
sp_help report on 262
resource limits
creating 44
dropping 210
information about 301
modifying 349
types of 44
retrieving
error message text 253
return status
catalog stored procedures 474
sp_checkreswords 107
system procedures 1
reversing encryption of source text 318
revoke auditing option 71
revoke option, **sp_role** 406
role hierarchies, displaying
using **sp_activeroles** 15
using **sp_displayroles** 189
roles
displaying with **sp_activeroles** 15
row lock promotion thresholds
dropping with **sp_droprowlockpromote** 213
setting with **sp_setrowlockpromote** 418
sp_helpdb report on database setting 278
row locks 334
rows, table
computing number of, with **sp_spaceused** 431
limiting how many returned 44
rpc auditing option 71
rpc security model A option, **sp_serveroption** 409
rpc security model B option, **sp_serveroption** 409
rules
binding 88–89
changing names of 110
checking name with **sp_checkreswords** 107
displaying source text of 314
naming user-created 88
remapping 393
renaming 398–399
system tables and 89
unbinding 464–465

S

scale, datatype
in user-defined datatypes 61
scope of resource limits
changes to active time ranges and 60
information on 301
specifying 46
security auditing option 72
security mechanism option, **sp_serveroption** 409

segments
  See also database devices; log segment; space allocation
  adding 49–50
  changing names of 112, 114
  checking names with **sp_checkreswords** 108
  dropping 214–215
  extending 50, 235
  information about 308
  mapping 50
  monitoring remaining space 54–57, 357–360
  sp_helpthreshold report on 316
select auditing option 73
server aliases 52
server cost option
  sp_serveroption 409
server information options. *See* information (server)
server process ID number. *See* processes (server tasks)
servers
  See also processes (server tasks); remote servers
  adding 51–53
  attribute names 486
  dropping 216
  information on remote logins 300
  local 51
  monitoring activity of 361
  names of 51
  options, changing with **sp_serveroption** 409–412
  remote 311
  setting row lock promotion thresholds for 418
  sp_server_info information on 486
  upgrading and **sp_checknames** 103
  upgrading and **sp_checkreswords** 107
set command
  sp_setlangalias and **language** option 413
setting
  auditing options 70
setuser auditing option 73
7-bit terminal, **sp_helpsort** output 312
shared locks 237, 333
shared row locks 335
single quotes. *See* quotation marks
single user database option
  setting with **sp_dboption** 163
single-user mode 163
  sp_renamedb and 400
size
  image datatype 430
  log device 339
  text storage 430
size of auto identity column configuration parameter 160, 164
sort order
  changing, and **sp_indsuspect** system procedure 322
  information about 312
source text
  checking for existence of 117
  displaying 314
  encryption, reversing 318
  hiding 318
sp_activeroles system procedure 15
sp_add_qpgroup system procedure 40
sp_add_resource_limit system procedure 44–48
sp_add_time_range system procedure 58–60
sp_addalias system procedure 16
sp_addauditrecord system procedure 17–18
sp_addaudittable system procedure 19
sp_addengine system procedure 21
sp_addexeclass system procedure 22
sp_addextendedproc system procedure 23–24
sp_addexternlogin system procedure 25–27
sp_addgroup system procedure 28
sp_addlanguage system procedure 29–31
sp_addlogin system procedure 32–34
sp_addmessage system procedure 35–36
sp_addobjectdef system procedure 37–39
sp_addremotelogin system procedure 41–43
sp_addsegment system procedure 49–50
  in mixed data and log databases 50
sp_addserver system procedure 51–53
sp_addthreshold system procedure 54–57
sp_addtype system procedure 61–64
sp_addumpdevice system procedure 65–66
sp_adduser system procedure 67–68
sp_altermessage system procedure 69
sp_audit system procedure 70–75
sp_auditdisplay system procedure 181–184
sp_autoconnect system procedure 76–77
sp_bindcache system procedure 78–81
sp_bindefault system procedure 82–83

create default and 83
sp_bindexeclass system procedure 84
sp_bindmsg system procedure 87
sp_bindrule system procedure 88–89
sp_cacheconfig system procedure 90–97
sp_cachestrategy system procedure 98–100
sp_changedbowner system procedure 101
sp_changegroup system procedure 102
  sp_dropgroup and 199
sp_checknames system procedure 103
sp_checkreswords system procedure 104–116
  return status 107
sp_checksource system procedure 117
sp_chgattribute system procedure 119–122
sp_clearpsexe system procedure 123
sp_clearstats system procedure 124–125
sp_cmp_all_qplans system procedure 128
sp_cmp_qplans system procedure 130
sp_column_privileges catalog stored procedure 477–478
sp_columns catalog stored procedure 479–480
  datatype code numbers 476
  and **sp_datatype_info** 482
sp_commonkey system procedure 132–133
sp_companion system procedure 134–136
sp_configure system procedure 137–142
  setting display levels for 185
sp_copy_all_qplans system procedure 143
sp_copy_qplan system procedure 144
sp_countmetadata system procedure 145
sp_cursorinfo system procedure 147–149
sp_databases catalog stored procedure 481
sp_datatype_info catalog stored procedure 482
sp_dbcc_alterws stored procedure 520
sp_dbcc_configreport stored procedure 521
sp_dbcc_createws stored procedure 522–523
sp_dbcc_deletedb stored procedure 524
sp_dbcc_deletehistory stored procedure 525–526
sp_dbcc_differentialreport stored procedure 527–528
sp_dbcc_evaluatedb stored procedure 529–530
sp_dbcc_faultreport stored procedure 531–533
sp_dbcc_fullreport stored procedure 534
sp_dbcc_plandb system procedure 380–381
sp_dbcc_runcheck stored procedure 536
sp_dbcc_statisticsreport stored procedure 537–539
sp_dbcc_summaryreport stored procedure 540–542
sp_dbcc_updateconfig stored procedure 543–545
sp_dbextend
  check 151
  clear 151
  database 151
  dbname 152
  devicename 152
  enable/disable 152
  freespace 152
  growby 151
  help 151, 152
  iterations 152
  list 151
  listfull 151
  maxsize 152
  modify 151
  newvalue 152
  permissions 156
  re-initializes sysattributes 152
  segmentname 152
  set 151
  simulate 151
  threshold 151
  trace 151, 152
  usage 154
  who 152
sp_dboption system procedure 157–164
sp_dbremap system procedure 167
sp_defaultloc system procedure 168–170
sp_depends system procedure 171–176
sp_deviceattr system procedure 177–178
sp_diskdefault system procedure 179–180
sp_displaylevel system procedure 185
sp_displaylogin system procedure 186–188
sp_displayroles system procedure 189
sp_drop_all_qplans system procedure 192
sp_drop_qpgroup system procedure 207
sp_drop_qplan system procedure 208
sp_drop_resource_limit system procedure 210–212
sp_drop_time_range system procedure 218
sp_dropalias system procedure 191
sp_dropdevice system procedure 193
sp_dropengine system procedure 194
sp_dropexeclass system procedure 195
sp_dropextendedproc system procedure 196
sp_dropexternlogin system procedure (Component Integration Services only) 197

sp_dropglockpromote system procedure 198
sp_dropgroup system procedure 199
See also **sp_changegroup**
sp_dropkey system procedure 200–201
sp_droplanguage system procedure 202
sp_droplogin system procedure 203
sp_dropmessage system procedure 204
sp_dropobjectdef system procedure (Component Integration Services only) 205–206
sp_dropremotelogin system procedure 209
sp_droprowlockpromote system procedure 213
sp_dropsegment system procedure 214–215
sp_placeobject and 214
sp_dropserver system procedure 216
sp_dropthreshold system procedure 217
sp_droptype system procedure 219
sp_dropuser system procedure 220
sp_dumpoptimize system procedure 221–225
sp_engine system procedure 226–228
sp_estspace system procedure 229–233
sp_export_qpgroup system procedure 234
sp_extendsegment system procedure 235
sp_familylock system procedure 237–239
sp_find_qplan system procedure 240–241
sp_fkeys catalog stored procedure 483–484
sp_flushstats system procedure 244
sp_forceonline_db system procedure 245
sp_forceonline_object system procedure 246–247
sp_forceonline_page system procedure 248–249
sp_foreignkey system procedure 250–251
sp_freedll system procedure 252
sp_getmessage system procedure 253
sp_grantlogin system procedure (Windows NT only) 254
sp_ha_admin system procedure 256
installing with *installhasvss* 256
sp_help system procedure 257–263
sp_help_qpgroup system procedure 297–298
sp_help_qplan system procedure 299
sp_help_resource_limit system procedure 301–303
sp_helpartition system procedure 264
sp_helpcache system procedure 267–268
sp_helpconfig system procedure 269–273
sp_helpconstraint system procedure 274–277
sp_helpdb and tempdbs 279
sp_helpdb system procedure 278–280
sp_helpdevice system procedure 281–282
sp_helpextendedproc system procedure 283
sp_helpexternlogin system procedure (Component Integration Services only) 284
sp_helpgroup system procedure 285
sp_helpindex system procedure 286–287
sp_helpjava system procedure 288–289
sp_helpjoins system procedure 290–291
sp_helpkey system procedure 292–293
sp_helplanguage system procedure 294
sp_helplog system procedure 295
sp_helpobjectdef system procedure (Component Integration Services only) 296
sp_helpremotelogin system procedure 300
sp_helprotect system procedure 304–307
sp_helpsegment system procedure 308–310
sp_helpserver system procedure 311
sp_helpsort system procedure 312–313
sp_helptext system procedure 314–315
sp_helpthreshold system procedure 316
sp_helpuser system procedure 317
sp_hidetext system procedure 318
sp_import_qpgroup system procedure 320–321
sp_indsuspect system procedure 322
sp_ldapadmin 323–325
sp_ldapadmin system procedure 323
sp_listener system procedure 326
sp_listsuspect_db system procedure 329
sp_listsuspect_object system procedure 330
sp_listsuspect_page system procedure 331
sp_lock system procedure 332–335
sp_locklogin system procedure 336–337
sp_logdevice system procedure 338–339
log on extension to **create database** and 338
sp_loginconfig system procedure (Windows NT only) 340–341
sp_logininfo system procedure (Windows NT only) 342
sp_logiosize system procedure 343
sp_modify_resource_limit system procedure 349–351
sp_modify_time_range system procedure 352–353
sp_modifylogin system procedure 346–348
sp_modifystats system procedure 354–356
sp_modifythreshold system procedure 357–360
sp_monitor

@@*connections* and 362
@@cpu_busy and 362
@@*pack_sent* global variable and **sp_monitor** 362
sp_monitor system procedure 361–363
sp_monitorconfig system procedure 364–370
sp_object_stats system procedure 371–373
sp_passthru system procedure 374–375
sp_password system procedure 376–377
sp_pkeys catalog stored procedure 485
sp_placeobject system procedure 378–379
sp_poolconfig system procedure 382–386
sp_primarykey system procedure 387
sp_foreignkey and 250
sp_processmail system procedure 388–389
sp_procxmode system procedure 390–391
sp_recompile system procedure 392–??
sp_remap system procedure 393
sp_remoteoption system procedure 394–395
sp_remotesql system procedure 396–397
sp_rename system procedure 398–399
sp_rename_qpgroup system procedure 402
sp_renamedb system procedure 111, 400–401
sp_reportstats system procedure 403–404
sp_revokelogin system procedure (Windows NT only) 405
sp_role system procedure 406
sp_sendmsg system procedure 407–408
sp_server_info catalog stored procedure 486–488
sp_tables and 497
sp_serveroption
external engine auto start 409
server cost 409
server costt 409
sp_serveroption system procedure 409–412
sp_set_qplan system procedure 417
sp_setlangalias system procedure 413
sp_setpglockpromote system procedure 414–415
sp_setpsexe system procedure 416
sp_setrowlockpromote system procedure 418
sp_setsuspect_granularity system procedure 420–422
sp_setsuspect_threshold system procedure 423
sp_showcontrolinfo system procedure 424
sp_showexeclass system procedure 426
sp_showplan system procedure 427
sp_showpsexe system procedure 429
sp_spaceused system procedure 430–432
sp_special_columns catalog stored procedure 489
sp_sproc_columns catalog stored procedure 491
datatype code numbers 476
sp_ssladmin system procedure 433–434
sp_statistics catalog stored procedure 493
sp_stored_procedures catalog stored procedure 495
sp_server_info information 487
sp_syntax system procedure 435–436
sp_sysmon system procedure 437–439
sp_table_privileges catalog stored procedure 496
sp_tables catalog stored procedure 497
sp_server_info information 487
sp_tempdb stored procedure
described 440
examples 442
parameters 440
syntax 440
usage 446
sp_thresholdaction system procedure 448–449
threshold procedure 55, 358
sp_transactions system procedure 450–456
sp_unbindcache system procedure 457–458
sp_unbindcache_all system procedure 459
sp_unbindefault system procedure 460
sp_unbindexeclass system procedure 461
sp_unbindmsg system procedure 463
sp_unbindrule system procedure 464–465
sp_volchanged system procedure 466–468
sp_who system procedure 469–472
columns returned 471
space
See also size; space allocation
estimating table and index size 229–233
monitoring remaining with **sp_modifythreshold** 357–360
sp_spaceused procedure 430–432
unused 431
space allocation
See also database devices; segments
future 378–379
log device 339
sp_placeobject procedure 378–379
space management properties
changing with **sp_chgattribute** 119
spid number
sp_who output 471

spt_committab table 5
spt_datatype_info table 475
spt_datatype_info_ext table 475
spt_monitor table 5
spt_server_info table 475
spt_values table 5
SQL standards
  SQL pattern matching 475
  user-defined datatypes and 62
square brackets [ ]
  in SQL statements xvii
starting days of named time ranges 58
starting times of named time ranges 58
statistics
  flushing to *systabstats* 244
  returned by global variables 361
  sp_clearstats procedure 124
  sp_monitor 361
  sp_reportstats 403–404
status
  database device 179
stored procedures
  See also database objects; system procedures
  cache binding and 80, 458
  catalog 473–497
  changing transaction modes with **sp_procxmode** 390–391
  for *dbccdb* database 517
  object dependencies and 171–176
  remapping 393
  renamed database and 401
  renaming 398–399
  sp_checkreswords and 108
  sp_recompile 392
  sp_sproc_columns information on 491
  sp_stored_procedures information on 495
suspect databases, listing 329
suspect indexes
  forcing online 246, 330
suspect pages
  bringing online 245, 248–249
  isolating on recovery 420–422, 423
  listing 331
sybdiagdb database 272
sybsyntax database 436
sybsystemprocs database
  permissions and 2
symbols
  in SQL statements xvi
syntax
  catalog stored procedures 474–475
  checking for reserved words 107
  display procedure (**sp_syntax**) 435–436
syntax conventions, Transact-SQL xvi
sysalternates table
  aliases 16
  sp_dropalias and 191
  sysusers table and 16
syscomments table
  source text in 315
sysconstraints table
  sp_bindmsg and 87
sysdatabases table 481
sysdevices table 179, 281
syskeys table
  sp_dropkey and 200
  sp_foreignkey and 250
  sp_primarykey and 387
syslanguages table 294
  sp_droplanguage and 202
syslkstats table 372
syslogs table 338
  put on a separate device 338
sysmessages table
  error message text 253
sysremotelogins table 41–43, 216
  sp_dropremotelogin and 209
sysresourcelimits table
  applicable limits for a login session 47
  sp_help_resource_limit and 302
sysservers table
  sp_addserver and 51
  sp_helpserver and 311
syssessions
  removing old entries 256
systabstats table
  flushing statistics to 244
system extended stored procedures 499–516
system procedure tables 5
  catalog stored procedures and 475
system procedures
  catalog stored 473–497

changing names of 110
displaying source text of 314
displaying syntax of 435–436
extended stored procedures 499–516
help reports 257–317
list of 1–14
permissions 2
return status 1
using 1
system procedures results. *See* information (server)
system roles
displaying with **sp_activeroles** 15
system segment
dropping 214
mapping 50
system tables
binding to caches 79
defaults and 83
direct updates dangerous to 112
rules and 89
space allocation 378
updating 1
systimeranges table
ID number storage in 59
range name storage in 44
systypes table 219
sysusermessages table
error message text 253
sp_dropmessage and 204
sysusers table
sysalternates table and 16

T

table_access auditing option 73
tables
binding to data caches 78
changing names of 110
checking name with **sp_checkreswords** 107
column information 479
column permission information from **sp_column_privileges** 477–478
common key between 132–133
dropping keys between 200
dropping row lock promotion thresholds for 213
estimating space for 229
joined common key 132–133
lock promotion thresholds for 415
locks held on 237, 333
locks, types of 237, 333
object dependencies and 171–176
primary keys on 387
renaming 398–399
setting row lock promotion thresholds for 418
sp_placeobject space allocation for 378–379
sp_recompile 392
sp_table_privileges information on 496
sp_tables 497
space used by 431
with suspect indexes 322
system procedure 5, 475
unbinding from data caches 457
tape dump devices
adding 65–66
tape option, **sp_addumpdevice** 65
tempdb
limiting number pages used per session 45, 350
tempdb database
auto identity database option and 160
unique auto_identity index database option and 164
tempdb_space 45, 350
tempdbs
sp_helpdb and 279
sp_tempdb descibed 440
sp_tempdb examples 442
sp_tempdb parameters 440
sp_tempdb syntax 440
sp_tempdb usage 446
temporary names. *See* alias, user
temporary tables
sp_help and 262
system procedure 5
terminals
7-bit, **sp_helpsort** output example 312
8-bit, **sp_helpsort** output example 312
text
copying with **defncopy** 109
user-defined message 35
text datatype
size of storage 430

@@thresh_hysteresis global variable
  threshold placement and 55
threshold procedures 55
  creating 448
  executing 56–57, 360
  parameters passed to 56, 359
thresholds
  adding 54–57
  changing 357–360
  crossing 54
  disabling 57, 217, 360
  hysteresis value 55, 358
  information about 316
  last-chance 55, 217, 358
  maximum number 55, 359
  optimization for reducing I/O 120
  removing 217
  row lock promotion 418
  space between 55
time interval
  estimating index creation 229
  limiting 45
  since **sp_monitor** last run 361
time ranges
  adding 58
  “at all times” 59, 218
  changing active time ranges 60
  creating 58
  dropping 218
  entire day 58
  IDs for 59
  modifying 352
  overlapping 59
timeouts option, **sp_serveroption** 410
@@total_errors global variable
  sp_monitor and 362
@@total_read global variable
  sp_monitor and 362
@@total_write global variable
  sp_monitor and 362
transaction logs
  data caches and 385
  log I/O size and 385
  on a separate device 338–339
  thresholds and 217
transactions
  modes 390–391
Transact-SQL
  reserved words 107
translation
  of user-defined messages 36
triggers
  changing names of 110
  checking name with **sp_checkreswords** 107
  displaying source text of 314
  object dependencies and 171–176
  remapping 393
  renamed database and 401
  renaming 398–399
  sp_recompile 392
true | false clauses
  sp_dboption 157
  sp_remoteoption 394
true option, **sp_changedbowner** 101
trunc log on chkpt database option 163
truncate auditing option 73
trusted mode
  remote logins and 43
trusted option, **sp_remoteoption** 394
two-phase commit
  probe process 404

U

UDP messaging 407
unbind auditing option 73
unbinding
  data caches 457–458
  defaults 460
  objects from caches 457–458
unencrypting source text 318
unique auto_identity index database option 164
unlocking login accounts 336
unmapping a segment from a database 214–215
unused space
  sp_spaceused reporting of 431
update auditing option 73
update row locks 335
us_english language 30
usage statistics 403
use message confidentiality server option 410

use message integrity server option 410
user context for operating system commands (**xp_cmdshell**) 501
User Datagram Protocol messaging 407
user IDs
  changing with **sp_import_qpgroup** 320
  displaying 187
  dropping with **sp_droplogin** and 203
user names
  See also database object owners; logins
  changing 112
  checking with **sp_checkreswords** 108
user permissions. *See* database owners; permissions
user-created objects. *See* database objects
user-defined audit records 70
user-defined datatypes
  binding defaults to 82–83
  binding rules to 88
  changing names of 111
  checking name with **sp_checkreswords** 107
  creating 61–64
  dropping 219
  hierarchy 63
  naming 63
  unbinding defaults from 460
  unbinding rules with **sp_unbindrule** 464–465
user-defined event logging (**xp_logevent**) 507
user-defined messages 35–36
  unbinding with **sp_unbindmsg** 463
user-defined procedures
  creating ESPs with **sp_addextendedproc** 23
user-defined roles
  displaying with **sp_activeroles** 15
users
  accounting statistics 124, 403
  adding 32–34, 67–68
  change group for 102
  changing names of 114, 346–348
  dropping aliased 191
  dropping from databases 220
  dropping from servers 203
  dropping remote 216
  information on 186, 317
  password change for accounts 376–377
  permissions of 304
  remote 300
  sp_who report on 469–472
  system procedure permissions and 2
  sysusers table 16
utility commands
  See also Utility Programs manual
  display syntax 435–436

V

values
  displaying with **sp_server_info** 486
view_access auditing option 73
views
  checking name with **sp_checkreswords** 107
  columns 479
  common key between 132–133
  displaying source text of 314
  dropping keys between 200
  object dependencies and 171–176
  primary keys on 387
  remapping 393
  renamed database and 401
  renaming 110, 398–399
virtual page numbers 282
volume handling 466

W

wash area
  configuring 385
  defaults 385
wash keyword, **sp_poolconfig** 382
weekday date value
  first 29
  names and numbers 29
wildcard characters
  SQL standards pattern matching ($ and _) 475
workspaces
  dropping 523

X

XP Server 500

freeing memory from 252
xp_cmdshell context configuration parameter 501
xp_cmdshell system extended stored procedure 501
xp_deletemail system extended stored procedure 504
sp_processmail and 389
xp_enumgroups system extended stored procedure 505
xp_findnextmsg system extended stored procedure 506
sp_processmail and 389
xp_logevent system extended stored procedure 507
xp_readmail system extended stored procedure 508
sp_processmail and 389
xp_sendmail system extended stored procedure 511
sp_processmail and 389
xp_startmail system extended stored procedure 515
xp_stopmail system extended stored procedure 516