This is an arbitrary selection of functionality which may be useful when working with ASE. All information applies to ASE 11.9.x or later, and to a Unix environment, except where indicated otherwise. *"(12.0)"* indicates new functionality in ASE 12.0.
Last updated: 25 November 2001. This document can be found at www.sypron.nl; to be notified of any updates, subscribe to the mailing list at www.sypron.nl/notify.html.
For a complete ASE Quick Reference Guide, covering all ASE functions, procedures and commands, go to **www.sypron.nl/qr**.

Contents

Datetime functions

Datetime formatting
Datetime and smalldatetime values can be formatted using the **convert** command: ***convert*** *(varchar, datetime_value, style).* Possible values for *style* are listed below. When *style* < 100, the digits for "century" are suppressed (except for style='0', which always displays the century). Without *style*, the default of '0' is used.

Style	Corresponding datetime formatting
0, 100	Apr 15 1999 5:41PM
1, 101	04/15/1999
2, 102	1999.04.15
3, 103	15/04/1999
4, 104	15.04.1999
5, 105	15-04-1999
6, 106	15 Apr 1999
7, 107	Apr 15, 1999
8, 108	17:41:52
9, 109	Apr 15 1999 5:41:52:283PM
10, 110	04-15-1999
11, 111	1999/04/15
12, 112	19990415

getdate()
Returns current date and time on the system where the ASE server is running. Note that this may be different from the date/time on the client system.

datename *(datepart, datetime_value)*
Returns the *datepart* (see below table) of a datetime value as an ASCII string.
Example: **select datename(dw, "15-Apr-1999")** returns "Thursday".

datepart *(datepart, datetime_value)*
Returns the *datepart* (see below table) of a datetime value as an integer.
Example: **select datepart(dw, "15-Apr-1999")** returns "5".

datediff *(datepart, datetime_value_1, datetime_value_2)*
Returns the difference between two datetime values (*datetime_value_2* - *datetime_value_1*) , expressed in the specified *datepart* unit (see below table).
Example: **select datediff(dd, "15-Apr-1998", "15-Apr-1999")** returns "365".

dateadd *(datepart, number, datetime_value)*
Returns the datetime value produced by adding *number datepart* units (see below table) to the datetime value specified. Specify a negative number to substract.
Example: **select dateadd(dd, 21,"15-Apr-1999")** returns "May 6, 1999 12:00 AM".

Possible values for *datepart* in the above functions are as follows:

Full name	Abbrev.	Values returned by datepart() & datename()
year	yy	1753 - 9999 (1753-2079 for smalldatetime)
quarter	qq	1 - 4
month	mm	1 - 12
week	wk	1 - 54
day	dd	1 - 31
dayofyear	dy	1 - 366
weekday	dw	1 - 7 (Sunday - Saturday)
hour	hh	0 - 23
minute	mi	0 - 59
second	ss	0 - 59
millisecond	ms	0 - 999
(12.0) calweekofyear	cwk	1 - 53
(12.0) calyearofweek	cyr	1753 - 9999 (1753-2079 for smalldatetime)
(12.0) caldayofweek	cdw	1 - 7 (Monday-Sunday)

Datatype conversion functions

convert *(datatype [(length)] | (precision[, scale])], expression[, style])*
Converts *expression* to *datatype*. *style* is used for datetime formatting only. *datatype* cannnot be a user-defined datatype. Example:
select convert(numeric(10,2), @float_value)
It is possible to specify a target datatype of **varchar** without specifying a length; this can be convenient for formatting purposes. Example:
select "[" + convert(varchar, 123) + "]" returns **"[123]"**

hextoint *(hex_string)*
Returns the platform-independent integer equivalent of a hexadecimal string, which must be either a character expression or a valid hexadecimal string, with or without a "0x" prefix, enclosed in quotes. Example: **select hextoint("0x1F")** returns **31**

inttohex *(integer)*
Returns the platform-independent hexadecimal equivalent of the *integer* expression, without a leading "0x". Example: **select inttohex(31)** returns **0000001F**

String functions

ascii (char_expr) - Returns the ASCII code for the first character in the expression.

char *(integer_expr)*
Converts a single-byte integer value to a single character. **char()** is the inverse of **ascii()**. *integer_expr* must be between 0 and 255. If the resulting value is the first byte of a multibyte character, the character may be undefined.
Examples: char(65)="A"; char(10) = newline; char(9) = tab.

charindex *(expression1, expression2)*
Searches *expression2* for the first occurrence of *expression1* and returns an integer representing its starting position. If *expression1* is not found, returns 0. If expression1 contains wildcard characters, these are treated as literals (also see **patindex()**).

char_length *(char_expr)*
Returns the number of characters in a character expression or text value. For variable-length data in a table column, **char_length()** strips the expression of trailing blanks before counting the number of characters. For multibyte character sets, use the system function **datalength()** to determine the number of bytes.

compare *(char_expr_1, char_expr_2, [collation_name | collation_nr])*
Compares the two character expressions using the specified collation (default is binary collation). Returns 1 if *char_expr_1* > *char_expr_2*; -1 if *char_expr_2* > *char_expr_1*; 0 if *char_expr_1* = *char_expr_2.*

difference *(char_expr1, char_expr2)*
Returns the integer difference between two soundex values. See **soundex()**, below.

lower *(char_expr)* - Converts the character expression to lowercase.

ltrim *(char_expr)* - Removes leading blanks from the character expression.

patindex *("%pattern%", character_expression [using {bytes | chars | characters}])*
Returns an integer representing the starting position of the first occurrence of *pattern* in *character_expression*; returns 0 if *pattern* is not found. By default, patindex returns the offset in characters. To return the offset in bytes (for multibyte charsets) specify *using bytes*. The % wildcard character must precede and follow *pattern*, except when searching for first or last characters. **patindex()** can be used on text data.

replicate *(char_expr, integer_expr)*
Returns a string containing *char_expr* repeated the *integer_expr* number of times or as many times as will fit into a 255-byte space, whichever is less. Example:
select replicate("xyz", 3) returns **"xyzxyzxyz"**

reverse *(expression)* - Returns the reversed character or binary *expression*. Example: **select reverse("abcd")** returns **"dcba"**

right *(expression, integer_expr)*
Returns the *integer_expr* right-most characters of the specified character or binary *expression*. Example: **select right("abcdefg", 3)** returns **"efg"**

rtrim *(char_expr)* - Removes trailing blanks from the character expression.

soundex *(char_expr)* - Returns a 4-character soundex code for character strings. Note: this is a rather rudimentary implementation of a soundex algorithm, which should not be mistaken for a full-fledged soundex-based retrieval system.

sortkey *(char_expr, [collation_name | collation_nr])*
Returns a varbinary(255) hex string representing the chracter expression according to the specified collation (default is binary collation).

space *(integer_expr)*
Returns a string containing *integer_expr* spaces. **space(0)** returns **NULL**.

str *(approx_numeric [, length [, decimal]])*
Returns a right-adjusted character representation of the floating point number. *length* (default=10) sets the number of characters to be returned (including the decimal point, all decimal digits, and blanks); *decimal* (default=0) sets the number of decimal digits to be returned. **str()** rounds the decimal portion of the number so that the results fit within the specified length.

stuff *(char_expr1, start, length, char_expr2)*
Delete *length* characters from *char_expr1* at *start*, and then insert *char_expr2* into *char_expr1* at *start*. To delete characters without inserting other characters, *char_expr2* should be NULL, not " ", which indicates a single space.

substring *(expression, start, length)*
Returns part of a character or binary string, starting at position *start* for *length* characters. Note that *length* may be specified longer than the remaining string length. Also note that by specifying 0 for *start*, a NULL value results.

upper *(char_expr)* - Converts the character expression to uppercase.

System functions

col_name *(object_id, column_id [, database_id])*
Returns the column name. Example: **select col_name(1,4)** returns “type”.

col_length *(object_name, column_name)*
Returns the defined length of column. Use **datalength()** to see the actual data size.

curunreservedpgs *(dbid, lstart, default_value)*
Returns the number of free pages on the device fragment starting at logical page number *lstart* (as stored in **master..sysusages**). *default_value* will be returned in case the information cannot currently be retrieved.

data_pgs *(object_id, {doampg | ioampg})*
Returns the number of pages used by the table (*doampg*) or index (*ioampg*), excluding pages used for internal structures. Use the function in a query against the **sysindexes** table; see the **rowcnt()** function for an example.

datalength *(expression)*
Returns the length of *expression* in bytes. *expression* is usually a column name. If *expression* is a character constant, it must be enclosed in quotes.

db_id *([database_name])* - Returns the ID number of the specified database. If no *database_name* is supplied, returns the ID number of the current database.

db_name *([database_id])* - Returns the database name for *database_id* (a numeric expression). If no *database_id* is supplied, returns the name of the current database.

host_id () / host_name ()
Returns the host process ID / computer name of the current session's client process.

index_col *(object_name, index_id, key_# [, user_id])* - Returns the name of the indexed column; returns NULL if *object_name* is not an exisiting table or view.

index_colorder *(object_name, index_id, key_# [, user_id])*
Returns the sort order of the indexed column, which is either "ASC" or "DESC" .

is_sec_service_on *(security_service_name)* - Returns 1 if the specified security service is active and 0 if not. Example: **select is_sec_service_on("unifiedlogin")**

isnull *(expression1, expression2)* - Substitutes the value specified in *expression2* when *expression1* evaluates to NULL; otherwise returns *expression1*. The datatypes of the expressions must convert implicitly, or you must use the convert function.

lct_admin *({ "lastchance" | "logfull" } , dbid)*
lct_admin *("abort" , spid [, dbid])*
lct_admin *("reserve", log_pages)*
Manages the log segment's last-chance threshold, or aborts suspended transactions:

- *"lastchance"* creates a last-chance threshold in the specified database.
- *"logfull"* returns 1 if the last-chance threshold has been crossed in the specified database or 0 if it has not.
- *"abort"* aborts the open transaction in session *spid.* If *spid* = 0, all open transactions in the specified database are aborted. Transactions will only be aborted when they are in "LOG SUSPEND" mode. (Note: **lct_admin("unsuspend")**, which existed in older ASE versions, is no longer available).
- *"reserve"* returns the number of free log pages required to successfully dump a transaction log of the specified size.

Example: **select lct_admin("abort", 19)**

(12.0) **license_enabled** *(option_name)* - Returns 1 if the specified option is licensed, and 0 if it is not. Example: **select license_enabled("ASE_JAVA")**

object_id *(object_name)*
Returns the object's ID, or NULL if the object doesn't exist. Examples:
select object_id("your_table")
select object_id("#mytable")
select object_id("tempdb..my_other_table")

object_name *(object_id [, database_id])*
Returns the object name, or NULL if it doesn't exist in the specified database.

ptn_data_pgs *(object_id, partition_id)*
Returns the number of data pages used by a table partition. Example:
select ptn_data_pgs(id, partitionid)
from syspartitions where id=object_id("my_table")

reserved_pgs *(object_id, {doampg|ioampg})* - Returns the number of pages allocated to table or index, including pages for internal structures. Use the function in a query against the **sysindexes** table; see the **rowcnt()** function for an example.

rowcnt *(doampg)*
Returns the number of rows in a table, without scanning the entire table; instead, the rowcount is retrieved from the table's OAM page. Note that this value will usually be correct, but may be slightly off during heavy insert/delete activity on the table. Use the function in a query against the **sysindexes** table, for example:
select rowcnt(doampg) from sysindexes
where indid<2 and id = object_id("my_table")

show_sec_services *()* - Lists active security services for the current session.

tsequal *(timestamp, timestamp2)*
Compares two timestamp values. This function is typically used in the WHERE-clause of an UPDATE-statement, to prevent update on a row that has been modified since it was selected for browsing. Example:
update mytable set column = column + 1
where primary_key = @value
and tsequal(ts_column, @stored_ts_value)

used_pgs *(object_id, doampg, ioampg)*
Returns the number of pages used by a table and its clustered index. Use in a query against the **sysindexes** table; see the **rowcnt()** function for an example.

valid_name *(character_expression)*

Returns 0 if the *character_expression* is not a valid identifier (illegal characters or more than 30 bytes long), or a number other than 0 if it is a valid identifier.

valid_user *(server_user_id)* - Returns 1 if the specified ID is a valid user or alias in at least one database on this ASE. You must have the **sa_role** or **sso_role** to use this function on a *server_user_id* other than your own.

Mathematical functions

abs *(number)* - Returns the absolute value of a given expression.

ceiling *(number)* - Returns the smallest integer greater than or equal to the specified expression. Example: **select ceiling(1.3)** returns **2**

exp *(number)* - Returns the exponential value of the specified value.

floor *(number)* - Returns the largest integer that is less than or equal to the specified value. Example: **select floor(1.9)** returns **1**

log/**log10** (number) - Returns the natural logarithm / base 10 logarithm of *number*.

pi *()* - Returns the constant value of 3.1415926535897931.

power *(value, power)* - Returns *value* to the power of *power*.

rand *([integer])*
Returns a random float value between 0 and 1, optionally using *integer* as a seed.

round *(number, integer)*
Rounds the *number* so that it has *integer* significant digits. A positive integer determines the number of significant digits to the right of the decimal point; a negative integer, the number of significant digits to the left of the decimal point.

sign *(number)* - Returns the sign of *number* : positive (+1), zero (0), or negative (-1).

sqrt *(number)* - Returns the square root of the specified value, which must be >= 0.

acos / **asin / atan** *(number)*
Returns the angle (in radians) whose cosine / sine / tangent is the specified value.

atn2 *(num1, num2)* - Returns the angle (in radians) whose tangent is (num1/num2).

cos / **cot** / **sin** / **tan** *(number)* - Returns the trigonometric cosine / cotangent / sine / tangent of the specified angle (in radians).

degrees *(number)* **/ radians** *(number)*
Converts radians to degrees / degrees to radians.

Bitwise operators

& - bitwise logical AND : **select 5 & 3** returns **1 (0101 & 0011 = 0001)**
| - bitwise logical OR : **select 5 | 2** returns **7 (0101 | 0010 = 0111)**
^ - bitwise logical EXOR : **select 5 ^ 3** returns **6 (0101 ^ 0011 = 0110)**
~ - bitwise logical NOT : **select ~1** returns **-2 (~0001 = 1110)**

Functions for "text" and "image" data

patindex *("%pattern%", char_expr [using {bytes | chars | characters}])*
See the **patindex()** string function.

textptr *(text_columname)*
Returns the text pointer value for the specified text/image column. Example:

```
declare @my_ptr varbinary(16)
select @my_ptr = textptr(text_col) from mytable
```

textvalid *("table_name.col_name", textpointer)*
Returns 1 if the given text pointer is valid, 0 if it's invalid. Note that the identifier for a text or image column must include the table name. Example:

```
select textvalid ("mytable.text_col", textptr(text_col)) from mytable
```

Login / User-related functions & commands

suser_id *([login_name])* - Returns the *suid* number (unique number for each login) from **master..syslogins** for the login name specified. If no login name is supplied, returns the *suid* of the current session.

suser_name *([suid])* - Returns the login name for the *suid* number specified. *suid* uniquely identifies a login as stored in **master..syslogins**. Without *suid*, returns the login name of the current session.

user
Returns the current session's database user name. Note: this is the only function that doesn't require brackets. Often used for column defaults in **create table** statements.

user_id *([user_name])*
Returns the database user's ID number for the specified user name, as stored in the **sysusers** table in the current database. If no *user_name* is supplied, it returns the ID of the current session's database user. Note: this should not be confused with server-wide login numbers as returned by the **suser_id()** function.

user_name *([user_id])*
Returns the database user's name for the specified user ID number. If no *user_id* is supplied, it returns the name of the current session's database user. Note: this should not be confused with server-wide login names as returned by the **suser_name()** function.

setuser *["user_name"]*
Changes the *uid* of the executing session to the *uid* of *user_name* in the current database. Without parameter, restores the original *uid*. Note: quotes are mandatory.

set proxy *login_name* / **set session authorization** *login_name*
Changes the *suid* of the executing session to the *suid* of *login_name,* thus obtaining all roles and permissions granted to *login_name*. To return to the original *suid*, specify the original *login_name*. Before this command can be used, a user having "sso_role" must execute **"grant set proxy to …"** from the master database. Note that granting this command may cause security problems. Note 2: **set session authorization** is ANSI-compliant; otherwise, both commands are identical.

sp_addalias *login_name, db_user_name*
Adds server login *login_name* as an alias of database user *db_user_name* in the current database. *login_name* must not already be a normal database user in this database. Alias definitions are stored in the **sysalternates** table.

sp_addgroup *group_name*
Adds a group (a named collection of database users) to the current database. Groups are stored in the **sysusers** table; a group's uid is >= 16384, except for the group "public," which has uid = 0.

sp_addlogin *login_name, passwd [, default_db [, default_language [, fullname]]]*
(12.0) **sp_addlogin** *login_name, passwd [, default_db [, default_language [, fullname [, passwd_exp [, min_pwd_len [, max_failed_logins]]]]]]*
Creates a new server login with the specified password and optional additional attributes (which can also be set through **sp_modifylogin**). Server logins are stored in the **master..syslogins** table.

sp_adduser *login_name [, db_user_name [, group_name]]*
Adds server login *login_name* as database user *db_user_name* in the current database. Without *db_user_name*, the user name will be equal to login_name. With *group_name*, also adds the new database user to this group. Database users are stored in the **sysusers** table. Database owner user **dbo** has uid=1; the **guest** user (if present) has uid=2.

(12.0) **sp_configure "systemwide password expiration",** *nr_of_days* (dynamic)
Defines the server-wide default number of days before a login or role password expires after being changed. Default = 0, meaning a password never expires.

(12.0) **sp_configure "minimum password length",** *nr_of_characters* (dynamic)
Defines the server-wide minimum number of characters for a login or role password.

(12.0) **sp_configure "maximum failed logins",** *nr_of_logins* (dynamic)
Defines the server-wide number of consecutive login (or "set role…on") failures before a login or role is locked. Default = 0, meaning this feature is disabled.

(12.0) **sp_configure "check password for digit",** *0 | 1* (dynamic)
Enables (1) or disables (0, =default) whether passwords must include a digit.

sp_changegroup *group_name, db_user_name*
Moves a database user into the group *group_name*, removing it from the group it was previously part of (if any). Note that a user will always be in the group "public". To remove a user from a group, run: **sp_changegroup "public",** *db_user_name*

sp_displaylogin *[login_name [, "expand_up" | "expand_down"]]*
Displays attributes of the specified server login. "*expand_up"* displays the parent roles of the login's roles; "*expand_down"* displays all roles contained in the login's

enabled roles. Note: to display all existing logins, use this query: **select name, suid from master..syslogins**

sp_dropalias *login_name* - Removes an alias database user name.

sp_dropgroup *group_name* - Removes a group, which must be empty.

sp_droplogin *login_name*
Drops the specified server login. Note that this will fail when the login is still mapped to any database user. In this case, download **sp_rv_helplogin** from `http://www.sypron.nl/new_ssp.html` to quickly fix the problem.

sp_helpgroup *[group_name]*
Displays information about one or all groups in the current database.

sp_helpuser *[db_user_name]*
Displays information about a database user(s), group, or alias in the current database. To display all aliases of the database owner, run: **sp_helpuser dbo**

sp_locklogin *[login_name, { **lock** | **unlock** }]*
Locks or unlocks a server login. Without parameters, displays all locked logins.

sp_modifylogin *login_name, option, value*
Modifies properties of a server login. Possible values for *option:*

- ***"defdb"*** - the default database in which the user will be located after login
- ***"deflanguage"*** - the official name of the user's default language
- ***"fullname"*** - the user's full name
- ***"add default role"*** - a user-defined role which has already been granted to this login, to be activated by default at login time
- ***"drop default role"*** - the reverse of ***"add default role"***
- ***(12.0) "passwd expiration"*** - number of days before the password expires, overriding the server-wide setting. If "0", the password never expires.
- ***(12.0) "min passwd length"*** - minimum number of characters in a password; when set to "0", blank passwords can be set.
- ***(12.0) "max failed_logins"*** - max. number of failed logins before the login gets locked. If "0", the login won't be locked.

(12.0) When specifying ***"passwd expiration", "min passwd length"*** or ***"max failed_logins"*** for *option*, and **"all overrides"** for *login_name*, the setting is applied to all logins with a non-default setting; when also specifying "-1" as *value*, all non-default settings will be dropped.

sp_password *caller_passwd, new_passwd [, login_name]*
Sets a new password for a login. *caller_passwd* is the executing login's current password; *new_passwd* is the new password being set. Only a login having **sso_role** can set another login's password; only in this case, *login_name* is used. Officially, you cannot set a password (back to) blank in pre-12.0; however, a solution can be found at `http://www.sypron.nl/blankpwd.html` .

Role-related functions & commands

proc_role *("role_name")* - Returns 1 if the current session has enabled the specified role directly; returns 2 if the role is contained in an enabled role; otherwise returns 0.

role_contain *("role_1", "role_2")* - Returns 1 if *role2* contains *role1*; otherwise returns 0. *role1* and *role2* are the names of system roles or user-defined roles.

role_id *("role_name")* **/ role_name** *(role_id)*
Returns the role ID / role name as stored in **master..syssrvroles.**

mut_excl_roles *(role_1, role_2 [**membership** | **activation**])*
Returns 1 if the two user-defined roles (or any roles directly contained in these roles) are defined as mutually exclusive, either on ***membership*** or ***activation*** level as specified; otherwise returns 0.

show_role *()*
Shows the session's currently active system-defined roles (not user-defined roles).

alter role *role_1* ***{ add | drop } exclusive { membership | activation }*** *role_2*
alter role *role_name* ***{ add passwd*** *new_password* ***| drop passwd }***
(12.0) **alter role** *role_name* ***{ add passwd*** *new_password* ***| drop passwd } {lock | unlock}***
(12.0) **alter role** *{ role_name* | ***"all overrides"*** *} **set { passwd expiration*** *nr_of_days* | ***min passwd length*** *min_length* | ***max failed_logins*** *max_number }*
Defines mutually exclusive relationships between roles and adds, drops, and changes the password for a role. In ASE 12.0, also locks or unlocks a role and sets

additional properties; when specifying **"all overrides"** for *role_name*, the setting is applied to all roles with a non-default setting; when also specifying "-1" as the setting value, all non-default settings will be dropped.

create role *role_name [**with passwd** new_password]*
(12.0) **create role** *role_name [**with passwd** new_password [, **passwd expiration** nr_of_days]]*
Creates a user-defined role. *role_name* must be unique in the server, and cannot be the name of an existing database user or group. When specifying a password, ASE users must specify this password to activate the role.
(12.0) passwd expiration defines the number of days before the password expires, overriding the server-wide setting. If "0", the password never expires.

drop role *role_name [**with override**]* - Drops the user-defined role *role_name*. Specify ***with override*** to also drop all associated permissions in all databases; otherwise, these permissions must be dropped first.

grant role *role_granted [, role_granted ...]* ***to*** *grantee [, grantee...]*
Grants roles to logins or roles. Roles cannot be granted to database user or groups. When granting a user-defined role to a login, by default it is not enabled at login time. This can be modified with **sp_modifylogin**.

revoke role *role_name [, role_name ...]* ***from*** *grantee [, grantee...]*
Revokes granted roles from logins or roles.

revoke *{ **all [privileges]** | command_list }* ***from*** *{* ***public*** | *name_list* | *role_name}*
Revokes permissions or roles from database users, groups or roles.

set role *role_name [**with passwd** password] }* ***{ on | off }***
Enables or disables r*ole_name* for the current session. Only roles that have been granted directly can be enabled; contained roles cannot be enabled independently.

sp_activeroles *["expand_down"]* - Displays enabled roles for the current session. "*expand_down"* displays all roles contained in the enabled roles.

(12.0) **sp_configure "systemwide password expiration",** *nr_of_days* (dynamic)
(12.0) **sp_configure "minimum password length",** *nr_of_characters* (dynamic)
(12.0) **sp_configure "maximum failed logins",** *nr_of_logins* (dynamic)
(12.0) **sp_configure "check password for digit",** *0 | 1* (dynamic)
See section on "Login/user-related functions&commands" for details.

sp_displayroles *[role_name | login_name [, option]]*
Displays all roles granted to a role or login (default = current session's login). Values for *option*: ***"expand_up"*** displays the parent roles; ***"expand_down"*** displays all roles contained in the enabled roles.
(12.0) "display_info" only shows info on *role_name*, but does not show if it is locked. To display all locked roles, use this query: **select name from master..syssrvroles where status&2=2**

sp_modifylogin *login_name, "{* ***add*** | ***drop*** *} **default role"**, role_name*
Enables/disables a user-defined role at login time. By default, a user-defined role is not enabled at login time. ***"add default role"*** enables the role. **sp_modifylogin** can only be run for roles that have been granted directly (i.e. not for contained roles).

sp_helprotect *@role_name* **=** *some_role*
Displays permissions granted to / revoked from *some_role* in the current database.

sp_role *{* ***"grant"*** | ***"revoke"*** *}, role_name, login_name*
Grants or revokes *role_name* to *login_name*. Equivalent to "**{grant | revoke} role** *role_name* **{to | from}** *login_name*"

Session-specific global variables

@@char_convert - Contains 1 if character set conversion is in effect; 0 if not.

@@client_csid / @@client_csname - Contains the ID / name of the client's most recently used character set. Contains -1 / NULL if the client charset hasn't been initialized.

(12.0) **@@curloid** - Contains the current session's LOID (Lock owner ID).

(12.0) **@@cis_rpc_handling** - Indicates whether outbound RPCs are handled by CIS (0=off=default; 1=on). Set through **set cis_rpc_handling *on|off***.

(12.0) **@@transactional_rpc -** Indicates whether RPCs are transactional (0=off=default; 1=on). Set through **set transactional_rpc *on|off*** (requires CIS).

(12.0) **@@textptr_parameters -** Indicates whether text/image data is passed "by value" through CIS RPC calls (0=off=default; 1=on). Set through **set textptr_parameters *on* | *off*** .

@@error - Contains the error status of the most recently executed statement in the current session. A value of 0 means successful completion; other values may indicate errors. Note: **@@error** is reset by commands that don't return rows, such as "if" statements, so **@@error** should be checked or saved before executing any other statements.

@@identity - Contains the last value inserted into an identity column in the current user session by an INSERT, SELECT INTO statement or a BCP operation. Following an insert into a table not containing an identity column, **@@identity** will be set to 0.

@@isolation - Contains the session 's current transaction isolation level (0, 1, 2 or 3). Note: level 2 is supported only for DOL tables; for APL tables, level 3 will be used.

@@langid / **@@language** - Contains the language ID / language name of the language currently in use, as specified in **syslanguages.langid**.

@@maxcharlen - Contains the max. length (in bytes) of multibyte characters in the default charset.

@@ncharsize - Contains the average length (in bytes) of a national character.

@@nestlevel - Contains the procedure nesting level in the current user session. This is initially 0, and is incremented each time a stored procedure or trigger is executed. The maximum allowed nesting level is 16.

@@options - Contains a bit map for the current status of some of the session's *set* commands. This variable is undocumented and bit mappings are platform-dependant. Check which bit(s) are affected on your platform for each *set* command.

@@parallel_degree / @@scan_parallel_degree - Contains the current maximum parallel degree setting for partition scans / hash-based scans.

@@procid - Contains the object ID of the currently executing stored procedure.

@@rowcount - Contains the number of rows affected by the last query. **@@rowcount** is reset by commands that don't return rows, such as "if" statements, so **@@rowcount** should be checked or saved before executing any other statements.

@@spid - Contains the server process ID number of the current session.

@@sqlstatus - Contains the result status of the last cursor **fetch** statement; other statements do not alter the value of **@@sqlstatus**. Possible values are:

- 0 - The fetch statement completed successfully.
- 1 - The fetch statement resulted in an error.
- 2 - End of result set reached.

@@tranchained - The session's current transaction mode: 0=unchained;1=chained.

@@trancount - Contains the nesting level of transactions in the current user session. This is incremented by each BEGIN TRANSACTION statement, and decremented by each COMMIT; a successful ROLLBACK (not to a savepoint !) resets it to 0.

@@transtate - Contains the state of the current transaction in the current user session. Values are:

- 0 - Transaction in progress
- 1 - Transaction successfully committed.
- 2 - Previous statement was aborted; transaction still in progress.
- 3 - Transaction aborted and rolled back.

@@textsize - Contains the max. number of bytes of text / image data that can be returned by a SELECT. The default depends on the client; for *isql*, the default (and maximum) is 32Kb. This setting can be changed for a session with **set textsize**.

@@textdbid / @@textobjid / @@textcolid - Contains the database ID / object ID / column ID of the object containing the column referenced by **@@textptr**.

@@textptr - Contains the text pointer of the last text or image column inserted or updated by a process (do not confuse this variable with the **textptr()** function).

@@textts - Contains the text timestamp of the column referenced by **@@textptr**.

Non-static server-wide global variables

@@connections - Contains the number of successful and attempted logins since ASE was last started.

@@cpu_busy / **@@io_busy / @@idle** - Contains the CPU / IO / idle time spent (in ticks) since ASE was last started. The length of a tick is stored in **@@timeticks**.

@@dbts - The value of the current database's "*timestamp"*, which is assigned to columns of the *"timestamp"* datatype. This platform-dependant value is internally incremented for every inserted, updated or deleted row, as well as for other reasons. Note: the **@@dbts** value cannot be related to the real-life date and/or time !

@@pack_received / @@pack_sent / @@packet_errors - Contains the number of network packets received / sent / in error since ASE was last started.

@@total_read / @@total_write / @@total_errors - Contains the number of disk reads / writes / errors since ASE was last started.

Static server-wide global variables

@@cis_version - Contains the date/version of CIS, if installed.

(12.0) **@@errorlog** - Contains the server errorlog pathname (undocumented).

@@max_connections - Contains the maximum number of concurrent connections that this ASE server can support in the current host computer configuration.

@@pagesize - Contains ASE page size (= 2048 bytes) (undocumented & useless).

@@servername - Contains the ASE server's name. To define **@@servername** as "MYSERVER" , run **exec sp_addserver "MYSERVER", "local"**, then shutdown & restart the server.

@@thresh_hysteresis - Contains the minimum amount of space (in 2Kb pages) that must be freed up above the threshold value before a threshold can be activated again. Also, thresholds on the same segment cannot be closer than twice this value.

@@timeticks - Contains the number of microseconds per tick. The amount of time per tick is machine-dependent, but usually 100000 (i.e. 100 milliseconds).

@@version - Contains the software version of the ASE server currently running.

Auditing-related commands (ASE 11.5 and later)

On Unix, auditing can be installed with the *auditinit* utility, but this can also be done manually as follows (on NT, only manual installation is possible in ASE 11.5+):

1. Create a database named **sybsecurity** with separate (this is important !) data and log devices. Recommended minimum size = 10Mb data + 2 Mb log.
2. Using *isql*, execute `scripts/installsecurity` (on NT: `scripts\instsecu`) in the ASE installation directory tree.
3. Shutdown and restart the server
4. Enable auditing with **sp_configure "auditing", 1**
5. (optional) Add additional audit tables with **sp_addaudittable** (see below)
6. Configure auditing settings with **sp_audit** (see below)
7. Query the system tables **sybsecurity..sysaudits_01..08** to view collected auditing records (see the *ASE Security Administration Guide* for details).

sp_configure "auditing" , *{ 0 | 1 }* - Enables (1) or disables (0) auditing (dynamic).

sp_configure "audit queue size", *size* - Sets the size of the audit queue (static config option); this is the max. number of audit records that can be cached.

sp_configure "current audit table" , *<n> [, "with truncate"]* (dynamic)
Sets the current audit table. <n> (0..8) specifies the new audit table. *"with truncate"* will truncate the new table first if it's not empty. 1..8 indicate sysaudits_01-08; 0 means the next in sequence. Default=1.

sp_configure "suspend audit when device full" , *{ 0 | 1 }* (dynamic)
Controls the behavior of the audit process when an audit device gets full: "0" switches to the next audit table, which is truncated first. "1"(=default) will suspend all audited operations (except for "sso_role" logins) until space becomes available.

sp_addauditrecord *[text [, db_name [, obj_name[, owner_name [, dbid [, objid]]]]]]*
Adds user-defined audit records (comments) into the audit trail.

sp_addaudittable *{ device_name | "**default**" }*
Adds a system audit table (auditing must already be installed), which is named **sysaudits_0x** (**x**=1..8). Each audit table will automatically be placed on its own segment which is created on the specified *device_name* where the new table will be placed onto (this device must exist in **sybsecurity**). With *"**default**"*, the new audit table and its segment are created on the first device of **sybsecurity**.

sp_displayaudit *["**procedure**" | "**object**" | "**login**" | "**database**" | "**global**" | "**default_object**" | "**default_procedure**" [, "**name**"]]*
Displays the status of auditing options set through **sp_audit**.

sp_audit *option, login_name, object_name [,setting]*
Configure auditing options (requires "sso_role"). *login_name* is a specific login to be audited. To audit all logins, specify **"all"** (including quotes!) for *option*; *login_name* can then be set to a system role (not a user-defined role) to audit all actions by users with that system role active. Display current settings with **sp_displayaudit**.
Values for *option* (evt=xx is the event code in the **sysaudits_0x.event** column):

- **"adhoc" -** allows adding ad-hoc audit records by **sp_addauditrecord** (evt=1)
- **"all" -** all actions performed by a particular user or by users with a particular role. Note: specifying **all** does not affect adding ad-hoc audit records
- **"alter" -** execution of the **alter table** (evt=3) or **alter database** (evt=2)
- **"bcp" -** execution of the **bcp-in** utility (evt=4)
- **"bind" - sp_bindefault** (evt=6)**, sp_bindmsg** (evt=7), **sp_bindrule** (evt=8)
- **"cmdtext" -** the T-SQL commands sent to the server by the client (evt=92) **"create" -** execution of **create database** (evt=9), **create table** (evt=10), **create procedure** (evt=11), **create trigger** (evt=12), **create rule** (evt=13), **create default** (evt=14), **sp_addmessage** (evt=15)**, create view** (evt=16)
- **"dbaccess" -** access to the current database from another database (evt=17)
- **"dbcc" -** execution of **dbcc** (evt=81)
- **"delete" -** deleting rows from a table (evt=18) or view (evt=19)
- **"disk" -** execution of **disk init**(evt=20), **disk refit**(evt=21), **disk reinit**(evt=22), **disk mirror**(evt=23), **disk unmirror**(evt=24)**, disk remirror** (evt=25)
- **"drop" -** execution of **drop database** (evt=26), **drop table** (evt=27), **drop procedure** (evt=28), **drop trigger** (evt=29), **drop rule** (evt=30), **drop default** (evt=31), **sp_dropmessage** (evt=32), **drop view** (evt=33)
- **"dump" -** execution of **dump database** (evt=34), **dump transaction** (evt=35)
- **"errors" -** fatal (evt=36) and non-fatal errors (evt=37)
- **"exec_procedure" -** execution of a stored procedure (evt=38)
- **"exec_trigger" -** execution of a trigger (evt=39)
- **"func_dbaccess", "func_obj_access"** - access to a database or database object via a T-SQL function (evt=85)
- **"grant" -** execution of the **grant** command (evt=40)
- **"insert" -** inserting rows into a table (evt=41) or view (evt=42)
- **"load" -** execution of **load database** (evt=43), **load transaction** (evt=44)
  "login" - all login attempts into ASE (evt=45)
- **"logout" -** all logout attempts from ASE (evt=46)
- **"reference" -** creation of **references** constraints between tables (evt=91)
- **"revoke" -** execution of the **revoke** command (evt=47)
- **"rpc" -** execution of RPC from (evt=48) or to another server (evt=49)
- **"security" -** following security-relevant events:
  - starting up (evt=50) or shutting down the ASE server (evt=51)
  - activating or deactivating a role (evt=55)
  - regenerating a password by an SSO (evt=76)
  - using the functions **valid_user()** (evt=85) or **proc_role()** (evt=80)
  - **sp_configure** (evt=82)
  - **online database** (evt=83)
  - running the commands **set proxy, set session authorization** (evt=88)
  - running the commands **connect to** (evt=90), **kill** (evt=89)
- **"select" -** selecting rows from a table (evt=62) or view (evt=63)
- **"setuser" -** execution of the **setuser** command (evt=84)
- **"table_access" -** access to any table by a specific user: **select** (evt=62), **delete** (evt=18), **insert** (evt=41), **update** (evt=70)
- **"truncate" -** execution of the **truncate table** command (evt=64)
- **"unbind" - sp_unbindefault** (evt=67), **sp_unbindrule** (evt=68), **sp_unbindmsg** (evt=69)
- **"update" -** updating rows in a table (evt=70) or view (evt=71)
- **"view_access"** - access to any view by a specific user: **select** (evt=63), **delete** (evt=19), **insert** (evt=42), **update** (evt=71)

object_name is the name of the object to be audited. Valid values are:

- the object name (may include owner name)
- **"all"** for all objects.
- **default table, default view** (when *option* is **delete, insert, select,** or **update)** for new tables or views
- **default procedure** (when *option* is **exec_procedure)** for new procedures
- **default trigger** (when *option* is **exec_trigger)** for new triggers

setting can be one of the following:

- **"on"** - Activates auditing for the specified option.
- **"off"** - Deactivates auditing for the specified option.
- **"pass"** - Activates auditing for events that pass permission checks.
- **"fail"** - Deactivates auditing for events that fail permission checks.

on and **off** apply to all options; **pass** and **fail** apply to all options except **errors** and **adhoc**. Specifying both **pass** and **fail** for the same option is equivalent to **on**.

Auditing-related commands (ASE 10.x & 11.0.x)

To install auditing, use "sybinit" (only on Unix) or do a manual installation as follows:

1. Create a database named **sybsecurity** with separate data and log devices.
2. (optional) Using **sp_addsegment**, create a segment in the **sybsecurity** database to place the **sysaudits** table onto.
3. From the **sybsecurity** database, execute **sp_auditinstall** (see below)
4. Using *isql*, execute `scripts/installsecurity` (on NT: `scripts\instsecu`) in the ASE installation directory
5. Execute **sp_auditoption "enable auditing", "on"** and restart the server
6. Configure auditing settings with **sp_audit*** procedures (see below)
7. Query the system table **sybsecurity..sysaudits** to view collected auditing records (see the *ASE Security Administration Guide* for details).

sp_configure "audit queue size", *size* - See description for ASE 11.5+ auditing.

sp_auditinstall *[segment [, device]]*
Creates the **sybsecurity..sysaudits** table on the (optional) *segment* and *device*.

sp_auditoption *[{"all" | "enable auditing" | "logouts" | "server boots" | "adhoc records"} [, {"on" | "off"}]]*
sp_auditoption *{"logins" | "rpc connections" | "roles"} [, {"ok" | "fail" | "both" | "off"}]*
sp_auditoption *"errors" [, {"nonfatal" | "fatal" | "both"}]*
sp_auditoption *"{sa | sso | oper | navigator | replication} commands" [, {"ok" | "fail" | "both" | "off"}]*
Enables and disables system-wide auditing and global audit options. The different options enable or disable the following auditing functions (note that the event codes are different from ASE 11.5+):

- *"enable auditing"* - enables (evt=1) / disables (evt=2) system-wide auditing.
- *"all"* - enables/disables all options except "*enable auditing" (*set this separately).
- *"logins"* - auditing of successful, failed, or all login attempts (evt=3).
- *"logouts"* - auditing of all logouts from the server (evt=4).
- *"server boots"* - generates an audit record when the server is rebooted (evt=5).
- *"rpc connections"* - audits incoming remote procedure calls (RPCs) (evt=6).
- *"roles"* - audits the use of the **set role** command (evt=7).
- *"errors"* - audits fatal errors, nonfatal errors, or both (evt=13).
- *"{sa|sso|oper|navigator|replication} commands"* - audits the use of privileged commands, requiring sa_role (evt=8), sso_role (evt=9), oper_role (evt=10), navigator_role (evt=12) or replication_role (evt=15) for execution.
- *"adhoc records"* - allows use of **sp_addauditrecord**.

sp_auditdatabase *[dbname [, "ok | fail | both | off"[, {"d u g r t o"}]]]*
Audits different types of events within a database (evt=100), or of references to objects within that database from another database. The event types ("d u g r t o") are *drop, use, grant, revoke, truncate* and *"outside access"*, respectively.

sp_auditobject *{objname | "default table"|"default view"}, dbname [, {"ok" | "fail" | "both" | "off"} [, "{d i s u}"]]*
Audits access to tables (evt=101) and views (evt=102). Specify *objname* to audit existing an table or view; use "*default table/view*" for tables/views that will be created in the future. Access types ("d i s u") are *delete, insert, select* & *update*, respectively.

sp_auditsproc *[sproc_name | "default" | "all", dbname [, {"ok" | "fail" | "both" | "off"}]]*
Audits the execution of stored procedures and triggers. Specify *sproc_name* to audit existing procedures (evt=103) or triggers (evt=104); use "*default*" for

procedures/triggers that will be created in the future. Use *all* for all procedure/triggers in the current database.

sp_auditlogin *[login [,"table" |"view" | "cmdtext" [, "ok" | "on" | "fail" | "both" | "off"]]]*
Audits a login's attempts to access tables (evt=105) and views (evt=106), or the text of commands that the user executes. For the *"cmdtext"* option (evt=107), only *"on"* and *"off"* apply.

sp_addauditrecord *[text] [, db_name] [, obj_name] [, owner_name] [, dbid] [, objid]*
Allows users to enter ad-hoc audit records (comments) into the audit trail (evt=14).

Setting up the "sybsyntax", "pubs2/3", "sybsystemprocs" databases

The **sybsyntax** database provides on-line syntax information on T-SQL commands and ASE system stored procedures. To install, create a database named **sybsyntax** (size 2Mb); then execute `scripts/ins_syn_sql` (in the ASE installation directory tree) using *isql*. You can then use **sp_syntax**. Examples:

sp_syntax "select" (note the quotes !)
sp_syntax sp_setpglockpromote

Note: **sybsyntax** is not available anymore in ASE 12.0.

The **pubs3** database is a small database on which all example queries in the Sybase documentation are based (**pubs2** is an older version). To install **pubs3** (or **pubs2**), execute `scripts/installpubs3 (or...pubs2)` (NT: `scripts\instpbs3 (or ...pbs2))` using *isql.* This will create a 2Mb **pubs3** (or **pubs2**) database on the default device (ASE 12.0: pubs3=3Mb). To create the database elsewhere, edit the script. The optional script `installpix2` (NT: `instpix2`) contains image data.

To manually (re)install the **sybsystemprocs** database, first create a database named **sybsystemprocs** of at least 60 Mb (mixed data and log is OK). Then execute `scripts/installmaster` (on NT: `scripts\instmstr)` using *isql.*

Setting up remote server CIS access

Server-to-server access through Component Integration Services (CIS, aka. "Omni") requires the following setup steps:

1. **sp_configure "enable cis", 1** (static option, requires server shutdown/restart)
2. in the local server, add the remote server name through **sp_addserver REMOTE_SERVER_NAME;** ensure the interfaces file used by the local server contains the correct interfaces definition for this remote server
3. set up external logins for connecting to the remote server using **sp_addexternlogin**.

sp_addexternlogin *remoteserver, local_login, extern_login [, extern_passwd]*
Defines (or modifies) CIS connection behaviour for the local server's *local_login*: connecting to *remoteserver* will be done as login *extern_login* using *extern_passwd.*
If no external login is defined for a local login, it will connect to the remote server as itself using its own password.

sp_dropexternlogin *remoteserver [, login_name]*
Drops external logins defined by **sp_addexternlogin**.

sp_helpexternlogin *remoteserver [, login_name]*
Displays information about external logins defined by **sp_addexternlogin**.

Setting up remote server RPC access

Server-to-server RPC access requires the following setup steps:

1. in both the local server (where the RPC is initiated from) and the remote server (where the executed RPC resides), define the **@@servername** through **sp_addserver MY_SERVER_NAME, "local"** ; then shutdown & restart the server and check **@@servername** is set correctly
2. ensure the config option **"allow remote access"** is set to 1 in both servers
3. in the local server, add the remote server's name through **sp_addserver REMOTE_SERVER_NAME** ; ensure the interfaces file used by the local server contains the corresponding interfaces definition for this remote server
4. in the remote server, set up a remote login to accept incoming RPC connections, using **sp_addremotelogin** (see below)
5. finally, execute an RPC from the local server to test the setup: **exec REMOTE_SERVER_NAME...sp_who**

sp_addserver *local_name [, class [, network_name]]*
Defines a remote server as *local_name* in the current server for server-to-server access. In the interfaces file, this server should exist as *network_name*. If *network_name* is omitted, it defaults to *local_name*. When *class* is "***local***", the local server's **@@servername** is defined as *local_name* (effective after a server restart). Otherwise, *class* should be NULL (see the *ASE Reference Guide* for more values).

sp_dropserver *servername [, "**droplogins**"]*
Drops the server definition for *servername* from **master..sysservers**. With *"**droplogins**"*, all associated remote logins are also dropped (by default, they're not).

sp_helpserver *[servername]*
Displays server definitions as stored in **master..sysservers** in the current server.

sp_addremotelogin *remoteserver [, loginame [, remotename]]*
Allows RPC logins coming from *remoteserver*. With only *remoteserver*, all remote logins are mapped to identical local logins. With *loginame*, all remote logins are mapped to the the local *loginame.* With *remotename*, that remote login is mapped to the local *loginame.* Note that the passwords for the logins must match in both servers.

sp_dropremotelogin *remoteserver [, loginame [, remotename]]*
Drops remote user login mappings established by **sp_addremotelogin.**

sp_helpremotelogin *[remoteserver [, remotename]]*
Displays remote user login mappings established by **sp_addremotelogin.**

sp_remoteoption *[remoteserver [,loginame [,remotename [,option [,"**true**"|"**false**"]]]]]*
Displays or defines options for remote server logins. Available options:

- *"**trusted**"* - when *"**true**"* (default is *"**false**"*), login passwords are not checked for the remote login mapping specified (note: this is a potential security problem !)

sp_serveroption *[remoteserver, option, "**true**" | "**false**"]*
sp_serveroption *[remoteserver, "**security mechanism**", mechanism_name]*
Displays or defines options for remote server connections. Possible *option* values:

- *"**net password encryption**"* - if "***true***", RPC connections with a remote server will use login password encryption (default is "***false***")
- *"**readonly**"* - if "***true***", access to the remote server is readonly (option only available with Component Integration Services)
- *"**timeouts**"* - when "false" (default is "true"), the site handler does not drop the physical connection after one minute with no logical connection
- ***"rpc security model A"*** - the default security model for handling RPCs, which does not support the features of security model B (see below)
- *"**rpc security model B**"* - a more advanced security model, supporting the features for authentication, confidentiality and integrity (see below), provided the underlying security mechanism supports these features.
- *"**mutual authentication**"* - valid with security model B only
- *"**use message confidentiality**"* - valid with security model B only
- *"**use message integrity**"* - valid with security model B only

Using the *"**security mechanism**"* option, a specific security mechanism can be specified through *mechanism_name*.

Setting up the "dbccdb" database

To run **dbcc checkstorage**, the **dbccdb** database must first be set up. Also, some configuration is required for each specific database that will be checked. Here's a summary of the steps required to prepare for a database named **my_db** :

1. From within the **master** database, run **sp_plan_dbccdb "my_db"** to obtain sizing and configuration recommendations (when omitting the parameter **"my_db"**, **sp_plan_dbccdb** will print recommendations for all databases).
2. If the **dbccdb** database doesn't exist yet:
   - create the **dbccdb** database on separate data and log devices according to the size recommendations. Note: make sure the log device is much larger than the recommended 2 Mb, otherwise it may be difficult to cleanup **dbccdb** later using **sp_dbcc_deletehistory**.
   - using *isql*, run the script `scripts/installdbccdb` (in the ASE installation directory tree)
3. Run **sp_configure "number of worker processes", "<number>"** to set this value at least as high as recommended by **sp_plan_dbccdb**, and restart ASE.
4. (Optional) Create a dedicated named cache for **dbcc checkstorage** (by default, the "default data cache" will be used) : **sp_cacheconfig "<cache-name>", "<size>M"**
5. In the cache used by **dbcc checkstorage**, configure a 16 Kb pool: **sp_poolconfig "<cache name>", "<size>M", "16K"**. <size> must be at least 640Kb*(recommended number of worker processes). As **dbcc checkstorage**

doesn't use the 2Kb pool, keep this at minimum size (512Kb) in a dedicated cache.

6. From within the **dbccdb** database, create scan and text workspaces (you may omit this step if you want to re-use already existing workspaces):
   sp_dbcc_createws dbccdb, "default", scan_my_db, scan, "<size>M"
   sp_dbcc_createws dbccdb, "default", text_my_db, text, "<size>M"
7. From within the **dbccdb** database, complete configuration issues (the values specified below should match the steps 3-6 above):
   sp_dbcc_updateconfig my_db, "max worker processes", "<number>"
   sp_dbcc_updateconfig my_db, "dbcc named cache", "<cache-name>", "<cache-size>"
   sp_dbcc_updateconfig my_db, "scan workspace", "scan_my_db"
   sp_dbcc_updateconfig my_db, "text workspace", "text_my_db"
   (further dbccdb configuration parameters exist, but are optional)
8. Now you're ready to run **dbcc checkstorage (my_db)**

After completing **dbcc checkstorage**, it is recommended to run **dbcc checkverify**. From within the **dbccdb** database, the results can then be displayed using **sp_dbcc_summaryreport** and **sp_dbcc_faultreport**. Periodically, the **dbccdb** configuration should be reviewed using **sp_dbcc_evaluatedb**.
Note that **dbcc checkstorage** can find corruptions, but cannot fix these; use **dbcc checkalloc / tablealloc / indexalloc** with the ***"fix"*** option for this. Fault code 100035 can be repaired by **dbcc checktable** with the ***"fix_spacebits"*** option.
When running **dbcc checkstorage** on the **dbccdb** database itself, the results will be recorded in the **dbccalt** database if it exists; if not, **dbccdb** will be used.

sp_dbcc_configreport *[db_name]* - Displays the **dbccdb** configuration settings used for all configured databases, or only for database *dbname*, when specified.

sp_dbcc_deletedb *[db_id | db_name]* - Deletes configuration settings from **dbccdb** for all configured databases, or only for the specified database.

sp_dbcc_deletehistory *[delete_until_datetime [, db_name]]*
Deletes results from **dbcc checkstorage** runs which completed before or on the date/time specified by *delete_until_datetime* from **dbccdb** for all configured databases, or only for *db_name*. Without a date/time, or by specifying NULL instead, all but the most recent set of results will be deleted.
Example: **sp_dbcc_deletehistory "15 April 2000", my_prod_db**

sp_dbcc_evaluatedb *[dbname]*
Re-computes recommended values for **dbccdb** configuration settings and compares these to the current settings. This is done for all configured databases, or only for database *dbname*, when specified. Note: **sp_dbcc_evaluatedb** doesn't show the configured named cache, but **sp_dbcc_configreport** does.

sp_dbcc_differentialreport *[dbname [, object_name]], ["**checkstorage**"] [, "date_1" [, " date_2"]]*
Displays the differences between the results of two **dbcc checkstorage** runs for all configured databases, or only for *dbname*, when specified. With *object_name*, only displays information for that object. With both dates, compares results from those dates. With one date, compares that date's result with the preceding result. Without dates, compares the two latest results.

sp_dbcc_faultreport *[report_type [, dbname [, object_name [, date]]]]*
Displays faults found by **dbcc checkstorage** for all configured databases, or only for database *dbname*, when specified. *report_type* can be ***"short"*** (brief output; default) or ***"long"***. With *object_name*, only displays faults for that object. With *date*, displays results for that date; otherwise displays most recent results.

sp_dbcc_fullreport *[dbname [, object_name [, date]]]*
Runs the following reports for all configured databases, or only for *dbname*: **sp_dbcc_summaryreport**, **sp_dbcc_configreport**, **sp_dbcc_statisticsreport**, and **sp_dbcc_faultreport "short"**. With *object_name*, only displays information for that object. With *date*, displays results for that date; otherwise displays most recent results.

sp_dbcc_runcheck *dbname [, final_procedure]*
Runs **dbcc checkstorage** and **dbcc checkverify** for the specified database. With *final_procedure*, runs this stored procedure after completion; otherwise runs **sp_dbcc_summaryreport** for *dbname*.

sp_dbcc_summaryreport *[dbname [, "**checkstorage**" | "**checkverify**"]]*

Displays a summary report from **dbcc checkstorage** and/or **dbcc checkverify** results (default=both) for all configured databases, or only for database *dbname*.

Supported DBCC commands

For most dbcc commands, trace flag 3604 must be enabled to see any output on the client. Most dbcc commands also require **sa_role** and/or **sybase_ts_role**.

dbcc checkalloc *[(dbname [, "fix" | "nofix"])]*
Checks and/or fixes corruptions in the specified database (default = current database). Default is *"nofix"* mode; for *"fix"*, the database must be in single-user mode. **dbcc checkalloc** is equivalent to **dbcc tablealloc** + **dbcc indexalloc** on all tables in the database.

dbcc checkcatalog *[(dbname)]* - Checks system table consistency in the specified database (default = current database). Does not fix any inconsistencies.

dbcc checktable *({ table_name | table_id } [, "skip_ncindex" | "fix_spacebits"])*
Checks (but does not fix) consistency of data and index pages for the specified table. To skip nonclustered indexes on user tables, specify *"skip_ncindex"*. To fix hard faults with code 100035 (from **dbcc checkstorage)**, use *"fix_spacebits"*.

dbcc checkdb *[(dbname [, "skip_ncindex" | "fix_spacebits"])]*
As **dbcc checktable**, but checks all tables in the specified database.

dbcc checkstorage *(dbname)*
Checks database consistence more effciently than the "classic" dbcc commands like **dbcc checkalloc/checkdb**. Requires that the **dbccdb** database be set up.

dbcc checkverify *(dbname)*-Reclassifies soft faults found by **dbcc checkstorage**.

dbcc dbrepair*(dbid | dbname, "dropdb")* - Drops a database marked suspect (when "drop database" can't); bits 256 and 64 must be in **sysdatabases.status**.

dbcc engine *("online")*
(12.0) dbcc engine *("offline" [, engine_nr])*
Dynamically starts ("online") or stops ("offline") a server engine. When specifying *engine_nr*, the corresponding engine is stopped; otherwise, stops the highest-numbered engine. The number of running engines must always remain between "min online engines" and "max online engines" (note that these config options are static). The *"online"* option can also be used in combination with traceflag 1608.
To display the current engine status: **select * from master..sysengines**

dbcc fix_text *(table_name | table_id)* - After changing to a new multi-byte character-set, this command must be run on all tables containing text columns.

(12.0) **dbcc complete_xact** *(xact_id, { "commit" | "rollback" })*
Heuristically completes (i.e. commits or rolls back) a distributed transaction.

(12.0) **dbcc forget_xact** *(xact_id)*
Removes the commit status of a heuristically completed distributed transaction.

dbcc gettrunc
Command to retrieve RepAgent/LTM information about the current ASE database.

- *ltm_truncpage* The first page that is not truncated in the database log
- *ltm_trunc_state* One of the following values:
  - 1 - ASE does not truncate the log on or after the ltm_truncpage
  - 0 - ASE ignores the ltm_truncpage
- *db_rep_stat* A mask constructed of the following:
  - 0x01 - ltm_truncpage is valid
  - 0x02 - database contains replicated tables
  - 0x04 - database contains replicated stored procedures
  - 0x20 - RepAgent enabled (ASE 11.5 / RS 11.5)
  - 0x40 - autostart RepAgent (ASE 11.5 / RS 11.5)
- *gen_id* The database generation ID
- *dbid* The ID number of the database
- *dbname* The name of the database
- *ltl_version* RepAgent: the log transfer language (LTL) version.
  LTM: the log transfer interface (LTI) version.

dbcc indexalloc *(table_name | table_id, index_id, [, report_type [,**"fix"** | **"nofix"**]])*
Checks and/or fixes corruptions in the specified index. *report_type* is ***"full"***, ***"optimized"*** *or* ***"fast"***(default=***"optimized"***). Default=***"fix"***, except for system tables.

(12.0) **dbcc rebuild_text** *(table_name | table_id [, col_name | col_id [, page]])*

(Re)creates the ASE 12.0 random-access structure for text/image data for a table, for a specific text/image column, or for the specified text/image page.

dbcc reindex *(object_name, object_id)*
Rebuilds a table's indexes (if necessary) after the changing the server's sort order.

dbcc settrunc *('ltm', {'valid' | 'ignore'})*
dbcc settrunc *('ltm', 'gen_id', db_generation)*
Command to modify the RepAgent/LTM information for the current ASE database.
Note: you cannot execute this command when RepAgent/LTM is running.

- *valid* - instructs ASE to respect the secundary truncation point.
- *ignore* - instructs ASE to ignore the secundary truncation point.
- *gen_id* - instructs ASE to reset the database generation number in the log (for primary databases).

d*b_generation* - is the new database generation number. Increment the number after restoring primary DB/log dumps to prevent RepServer from rejecting new transactions as duplicates.

dbcc tablealloc *(table_name* | *table_id [, report_type [,* ***"fix"*** | ***"nofix"*** *]])*
Checks and/or fixes corruptions in the specified table and its indexes. *report_type* is ***"full"***, ***"optimized"*** *or* ***"fast"***(default=**"*** ***optimized"***). Default=***"fix"***, except for system tables.

dbcc textalloc *(table_name* | *table_id [, report_type [,****"fix"*** | ***"nofix"*** *]])*
Checks and/or fixes corruptions in the text and image columns for the specified table. *report_type* is ***"full"***, ***"optimized"*** *or* ***"fast"*** (default = **"*** ***optimized"***). Default is ***"fix"***.

dbcc traceon | **traceoff** *[(traceflag [, traceflag…])]*
Enable / disable server trace flags. If no traceflags are specified in **dbcc traceoff**, all previously enabled flags are disabled.

dbcc tune *(option, value_1, value_2…)*
Modify certain run-time configuration settings. These setting are not persistent (i.e. not maintained after a server restart) unless specified otherwise. Possible options:

- *("ascinserts", value, table_name)* - if *value* = 0, ascending inserts are disabled (=default) for the specified table; if *value* = 1, enabled. This setting is persistent (stored in bit 64 of **sysindexes.status2**).
- *("cleanup", value)* - if *value* = 0, memory cleanup checking is enabled (=default); if *value* = 1, disabled.
- *("cpuaffinity", firstcpu , "on"* | *"off")* - enables/disables CPU affinity if supported by the platform; starting with engine 0, engines are bound to CPU *firstcpu.* Running ("cpuaffinity", -1) writes current setting to server errorlog.
- *(12.0) ("des_greedyalloc", dbid, table_name, "on"* | *"off")* - (re)sets a "greedy allocation" scheme for the table (to help reduce latch contention).
- *("deviochar", vdevno, value)* - sets maximum of *value* (1..255; default=3) outstanding I/O's by housekeeper for device *vdevno*; when *vdevno* = -1; applies to all devices
- *("doneinproc", { 0* | *1 })* - enables (1,=default) or disables (0) suppression of most *doneinproc* network packets. To disable all *doneinproc* network packets, boot the server with traceflag 292, and run *dbcc tune("doneinproc",0).*
- *("maxwritedes", value)* - sets max. number of outstanding disk writes (1..50; default=10)

Unsupported DBCC commands

WARNING ! These (arbitrarily selected) DBCC commands represent unsupported functionality which may cause irreversable damage to your databases and/or lead to data loss. Use entirely at your own risk. Do not contact Sybase Technical Support for assistence !

Many dbcc commands require that the role **sybase_ts_role** be enabled for the executing session. This can be done as follows (assuming the login is "sa"):

```
grant role sybase_ts_role to sa
set role sybase_ts_role on
```

dbcc help *(command)* - Displays syntax info for the specified dbcc command.

dbcc allocdump *(dbid* |*dbname, alloc_pageno)* - Lists extents for an allocation unit.

dbcc bytes *(address, length)*
dbcc bytes *(address, printopt, structure_name)*
Dumps *length* bytes, starting at *address* (e.g. a physical page address), in hex & ascii. **dbcc bytes***(0,0,"showlist")* displays all structures that can be specified as masks in **dbcc bytes***(address, printopt, structure_name).* Specify -1 for *printopt*.

dbcc cis *("subcommand")*
Displays CIS-related information. Subcommands: "*remcon"* displays all CIS remote connections; "*rusage"* - displays CIS shared memory usage. *"srvdes [, srv_id]"* - without argument, displays all SRVDES structures. With a server-id as argument, syncs the in-memory version of SRVDES with **master..sysservers**.

dbcc corrupt *(table_name* | *table_id, index_id, error_nr)*
Creates a corruption of the specified *error_nr* in the specified table or index. The table must reside in, and this command must be issued from, a database named "victimdb". Possible *error_nr* values are 1133, 2502, 2503, 2521, 2523, 2525, 2529, 2540, 2546, 7939, 7940, 7949.

dbcc dbinfo *(dbname)* - Displays the DBINFO structure for the specified database.

dbcc dbrecover *(dbid* | *dbname)*
Performs recovery for the specified database (without restarting the ASE server); works only when bit 64 in **sysdatabases.status** is set.

dbcc dbrepair*(dbid* | *dbname, option [, table_name, index_id])*
Performs various maintenance actions. Possible *option* values:

- *"dropdb"* - drops a database marked suspect (when "drop database" can't); bits 256 and 64 must be set in **sysdatabases.status**.
- *"findstranded"* - displays object extents that are "lost" on a logsegment
- *"ltmignore"* - like **dbcc settrunc(ltm, ignore)**, but works for an offline database: can be issued from outside the target database.
- *"remap"* - refreshes in-memory copy of **master..sysusages**
- *"repairindex"* - rebuilds a system table index (dangerous!!). The database must be in single user mode and **sysobject.sysstat** must have bit 4096 set.

dbcc dbtable *(dbid* | *dbname)*
Displays the DBTABLE structure for the specified database (includes "keep count")

dbcc des *[(dbid* | *dbname [,object_name* | *object_id])]*
Displays the DES structure for the specified object(s)

dbcc extentcheck *(dbid, table_id, index_id, sort_bit)*
Displays all extents of the specified table or index with the "sort bit" specified (0 or 1).

dbcc extentdump *(dbid, page_no)* - Displays the specified page's extent.

dbcc extentzap *(dbid, object_id, index_id, sort_bit)*
Delete extents for the specified object and index with the "sort bit" specified (0 or 1). To delete an object completely, run once with *sort_bit* = 0 and once with *sort_bit* = 1.

dbcc findnotfullextents *(dbid, object_id, index_id, sort_bit)*
Finds extents for the table or index which contain unused pages; s*ort_bit* is 0 or 1.

dbcc fix_al *[(db_name)]*
Fixes allocation errors within the specified database (default = current database).

(12.0) **dbcc istraceon** *(traceflag)*
Tests if the specified traceflag is enabled; if so, **@@error** will be 0; otherwise <> 0.

dbcc listoam *(dbid* | *dbname, object_name* | *object_id, index_id)*
Displays the OAM pages for the specified object or index.

dbcc locateindexpgs *(dbid, object_id, page_nr, index_id, level)*
Within the specified index on the specified index *level*, finds all references to the specified *page_nr* (*level* should be 0 when *page_nr* is a data page).

dbcc lock - Displays currently active locks.

dbcc log *([dbname* | *dbid] [, objid] [, page] [, row] [, nr_recs] [, type] [, printopt])*
Displays contents of the transaction log for the specified database. Parameters:

- if *objid* < 0, the log is scanned from *page* and *row* onwards; if *objid* < 0 and *page* = -3, shows log records for spid = abs(*objid*); if *objid* > 0 and *page* > 0, shows all log records for the transaction-ID formed by *page* and *row* ; if *objid* > 0 and *page* = *row* = 0, shows all log records for the object *objid*; if *objid* = 0 and *page* > 0, shows modifications to database page *page*; if *objid* = 0 and *page* = *row* = 0, shows all log records for the specified database.
- *row* should be set to 0 if not used.
- if *nr_recs* = 0, shows all log records; if *nr_recs* > 0, shows the first *nr_recs* matching log records. If *nr_recs* < 0, shows the last *nr_recs* matching log records in reverse order.

- if *type* = -1, shows all log records; If *type* >= 0, shows only log records of the specified type(example: 0=begin transaction; 30=commit/rollback; 17=checkpt).
- if *printopt* = 1, shows only log record headers. If *printopt* = 0, shows complete log records (=default).

Example: to find the last checkpoint in database 9: **dbcc log(9,0,0,0,-1,17,0)**. To dump the last 1000 log records in backward order: **dbcc log(9,0,0,0,-1000,-1,0)**

dbcc logprint *("string" [, "string" …])* - Writes the specified text strings to the server errorlog. Ignore the text "background task error -1:" that shows up in the errorlog.

(12.0) **dbcc printolog** *("string")*
Prints the specified text string to the server errorlog.

dbcc logtransfer *(…)* - Used by LTM or RepAgent. Not for manual use.

dbcc memusage - Displays information on internal ASE memory usage.

dbcc monitor *(…)*
Used by **sp_sysmon** to fill the **sysmonitors** table. Not for manual use.

dbcc object_atts *(object_name, atribute_nr, option [, new_value])*
Displays or patches data attributes on the OAM page of the specified object in the current database. Currently, this only supports identity values, for which *attribute_nr* must be 0. *option* can be "*get" (*displays the current value), "*reserve" (*reserves the value), or *"put"* (puts *new_value* (must be hexadecimal !) into the OAM page). Note: this command can be difficult to use !

dbcc object_stats *(action [,dbid | dbname [,object_id | object_name]])*
Displays lock wait times for a specific database and/or a specific table (default = all tables in all databases). Before running this command, enable trace flag 1213 first. *action* can be *"init_locks"* (to clear the counters) or *"locks"* (to display lock wait times). Also used by **sp_object_stats**.

dbcc page *(dbid | dbname, page [, printopt [, cache [, logical [, cache_name]]]])*
Dumps page *page* within the specified database. *printopt* can be 0 (=default; print page header), 1(print page header, data rows and row offset table), 2 (print page header and hex dump of data page) or 3 (print the control page for table partition). When *cache* = 1, the page is read from cache (default); if 0, it's read from disk. If *logical* =1, *page_nr* is a logical page. If 0, a virtual page (*cache* must be 0). If *cache_name* is -1, seaches in all caches; if NULL (=default), searches in cache bound to object; otherwise searches in *cache_name*.

dbcc pglinkage *(dbid | dbname, start_pg, nr_pgs, printopt, search_pg, order)*
For APL tables, walks down a page chain, starting at *start_pg*, for *nr_pgs* pages. If *nr_pgs* = 0, walks until end of chain or until *search_pg* is found (to search until end, set *search_pg* = 0). *printopt* can be *0* (shows number of pages accessed), *1* (displays page numbers of last 16 pages), or *2* (displays page numbers of all pages accessed). If *order* = 0, walks backward; if 1, walks forward.

dbcc procbuf *([dbid | dbname [,object_id | object_name [, nbufs [, printopt]]*
Displays query plans (proc. buffers/headers) in the procedure cache. By default, the entire proc.cache is displayed; specify a database and/or object as a search filter. When *nbufs* > 0, the first *nbufs* matching entries are displayed in MRU-LRU order; when < 0, in LRU-MRU order. When *printopt* = 0 (default), only headers are displayed; when 1, displays details. Note: the output can to be voluminous.

dbcc prtipage *(dbname, object_name, index_id, index_page_nr)*
Prints all index or data pages pointed to by the index rows on index page *index_page_nr* of the specified index.

dbcc pss *[([suid [, spid [, printopt]])]*
Prints PSS info for selected sessions; specify *suid*=0 to indicate a specific *spid*. *printopt* can be 0 (displays basic PSS structures), 1 (additional details), 2 (locks), 3 (cursors), 4 (active transactions) or 5 (worktables). The (last part of the) session's currently executing T-SQL command can often be displayed using **dbcc pss** (but better use **dbcc sqltext** for this).

dbcc rebuild_log *(dbid | dbname, rebuild_flag, delete_flag)*
Creates a new, "empty" (containing only a checkpoint record) transaction log for the database. Requires **sysdatabases.status** = -32768. Specify "1" for *rebuild_flag* and *delete_flag*. Note: this will significantly increase the value of the database's timestamp (so don't run this command too often).

dbcc rebuildextents *(dbid, object_id, index_id)*

Rebuilds the extents and OAM pages for the specified object and index.

dbcc resource - Displays internal settings; the errorlog location is shown at *"rerrfile".*

dbcc save_rebuild_log *(dbid | dbname)*
Same as **dbcc rebuild_log** but also saves the old log extents with object_id = -1.

dbcc sqltext *(spid)*
Displays the last 200-or-so characters of the currently executing / most recently executed T-SQL command for the specified session (like **dbcc pss**, but brief). Note: you can also use **sp_showplan** to display a spid's currently executing query's plan.

dbcc stacktrace *(spid | -kpid)*
Prints a stacktrace for the specified *spid* (positive value) or *kpid* (negative value).

dbcc traceflags *[(printopt)]*
Display traceflags currently enabled in the server, when *printopt* is 1 or omitted. When *printopt* is 2, also displays which sessions have enabled any traceflags.

dbcc thresholds *(dbid | dbname)* - Displays a database's threshold cache.

dbcc usedextents *(db_name | db_id, type, printopt, [bypiece])*
Displays extents used by the entire database (*type*=0) or just the log devices (*type*=1). When *printopt*=0, displays extent details; when 1, display count. When *bypiece*=1, output is by device fragment; when 0, it's not (=default)

dbcc user_stats *(action, spid)*
Displays how long a specific spid had to wait for locks. When action = *"init_locks"*, counters are cleared; when action = *"locks"*, displays results. If *spid* = 0, show results for all spids. Before running this command, enable trace flag 1213 first.

dbcc xls *("plcstate" | "plcflush" , spid)*
Operates on the User Log Cache (a.k.a. Private Log Cache or PLC) of session *spid*; when *spid*=0, applies to all sessions:

- *"plcstate"* - Displays the contents of the ULC
- *"plcflush"* - Flushes the ULC to the transaction log

Trace flags

WARNING ! These (arbitrarily selected) trace flags represent unsupported functionality which may cause irreversable damage to your databases and/or lead to data loss. Use entirely at your own risk. Do not contact Sybase Technical Support for assistence !

Trace flags can be set through **dbcc traceon/traceoff** or added to the RUN_SERVER file (i.e. "-T292"), depending on their function and the desired effect.

292 - reduces network traffic by suppressing server-to-client "done-in-proc" network packets in combination with **dbcc tune("doneinproc",0)**. (Note: in previous versions of this QuickRef, trace flag 260 was mistakenly mentioned here; however, 260 was for ASE version 10, and has been replaced with trace flag 292).
299 - don't recompile stored procs inheriting a temp table from a parent proc
302 - print info about the optimizer's index selection
310 - print info about the optimizer's join selection (for better join orders)
317 - print info about the optimizer's join selection (for rejected join orders)
319 - print info on reformatting decisions during query processing
1204 - replaced by config option "print deadlock information"
1213 - used with **dbcc object_stats**; see there
(12.0) **1608** - starts only engine 0 ; for use with **dbcc engine** *("online")*
1610 - replaced by config option "tcp no delay"
2512 - **dbcc checkalloc** will skip **syslogs** to suppress spurious alloc. errors
3300 - prints detailed info when performing recovery (can be voluminous)
3502 - log checkpointing of databases in the server errorlog
3604 - redirect dbcc output to client session
3605 - redirect dbcc output to the server errorlog
3607 - opens only master device; "master" db is accessible but not recovered
3608 - recovering only "master" db; other databases are accessible but not recovered; to avoid "log suicide", use "shutdown with nowait" and do not checkpoint.
3609 - all databases are recovered, but tempdb is not re-initialised
3610 - divide-by-zero results in NULL (as in version 4.x), instead of an error
4001 - write every login record to the server errorlog
4012 - don't start checkpoint process
4013 - write loginname, spid and kpid to the server errorlog for every login
4044 - allows to log into server as "sa" when this login is locked
8003 - prints info on RPC calls

8203 - display locking info when deadlock occurs
8399 - used by **dbcc monitor** to fill **sysmonitors** table
11201 - Logs CIS client connect- / disconnect- and attention events
11202 - Logs CIS client language, cursor declare, dynamic prepare, and dynamic execute-immediate text
11203 - Logs CIS client rpc events
11204 - Logs all CIS messages routed to client
11205 - Logs all CIS interaction with remote servers
11209 - For "update statistics" on proxy tables, only get row counts, not complete distribution statistics
11218 - Cursors involving proxy tables will be "read only" by default

New locking-related commands in ASE 11.9.x

ASE 11.9.x introduces data-only locking (DOL) as well as new locking commands. Allpages locking (APL), as used in pre-11.9, is still the default lock scheme. Below are the main related commands.

sp_configure "lock scheme", *0, { **allpages** | **datarows** | **datapages** }* (dynamic)
Sets the default server-wide lock scheme; "allpages" is the default. "datarows" and "datapages" are often abbreviated as DOL (data-only locking), and "allpages" (a.k.a. page locking) as APL. "Datarows" locking is also known as "row-level locking".

create table *table_name (column_list) [**lock** { **allpages | datarows** | **datapages**}]…*
Creates a table with the specified lock scheme. If none is specified, the server-wide default is used. A table's lock scheme is stored in **sysobjects.sysstat2** as follows:

sysobjects.sysstat2 bit set	lock scheme
bit 8192 (0x2000)	allpages
bit 16384 (0x4000)	datapages
bit 32768 (0x8000)	datarows
None of the above	allpages

alter table *table_name* ***lock*** *{ **allpages** | **datarows | datapages** }*
Changes the lock scheme for the specified table. For large tables, changing between APL and DOL schemes may take a long time, as the table is converted. Changing between datarows and datapages schemes is fast, irrespective of the table size. Note: changing between APL and DOL schemes is a "minimally logged" operation. Note2: when changing between APL and DOL schemes, best drop and recreate any triggers on the table (especially in ASE 11.9.x).

select *select_list* **into** *table_name [**lock** { **allpages | datarows |datapages** }] …*
Creates a table through **select..into** with the specified lock scheme. If none is specified, the server-wide default is used; not the lock scheme of the source table(s).

sp_configure "lock wait period" *, number_of_seconds* (dynamic)
Sets the server-wide lock-wait period; 0 means "nowait". To restore the default of "wait forever", run **sp_configure "lock wait period" , 0, "default"**.

set lock *{ **wait** [nr_secs] **| nowait** }*
Sets the lock wait timeout (in seconds) for the current session or in a stored procedure, which overrides a server-wide level setting. **set lock wait** (without *nr_seconds*) will wait forever (which is the default); **set lock nowait** is identical to **set lock wait 0**, and will not wait at all. When a query cannot acquire a lock within the specified time, **@@error** will be set to 12205 and the transaction will be aborted. Note: once **set lock** has been used in a session, the server-wide default has been overridden for the rest of this session.
Example: **set lock wait 60** sets the lock wait period for the current session to 1 minute.

lock table *table_name **in** { **share** | **exclusive** } **mode** [**wait** nr_secs **| nowait**]*
Within a transaction, explicitly locks a user table in the specified mode until the transaction completes. A shared lock can be upgraded to exclusive mode by re-issuing the command. When the lock cannot be acquired within the specified time, **@@error** will be set to 12207, but the transaction will continue.

{ **select | insert | update | delete** } *… **from** table_name [**readpast**]*
For DOL tables only, specifying **readpast** will cause the executing command not to be blocked by incompatible locks; instead, these locks will be skipped, so that the command can continue. **readpast** is also available with **readtext** and **writetext**.

set transaction isolation level *{ **2 | repeatable read** }*
For DOL tables only, transaction isolation level 2 ("repeatable read") is supported. When specified for an APL table, isolation level 3 ("serializable") will be used instead.

sp_configure "read committed with lock", *0 | 1* (dynamic)
For DOL tables only, determines whether SELECT queries on transaction level 1 and read-only cursors will hold shared row/page locks. Default = 0 = disabled.

sp_object_stats *"hh:mm:ss" [, top_N [, dbname [, object_name [, option]]]]*
During the specified period, collects locking statistics on server-, database- or object-level. When specified, displays only the *top_N* objects. *option* can be *"rpt_objlist"* (displays only object names) or *"rpt_locks"* (=default; displays details).

Lock promotion thresholds

Lock promotion decisions involve the following configurable thresholds (note: the below description is for APL tables; for DOL tables, read "row" instead of "page"):

- **LWM** (Low Water Mark; must be >=2; default=200) - when the number of page locks on a table is below the LWM, no promotion to a table lock will occur
- **HWM** (High Water Mark; default=200) - when the number of page locks on a table exceeds the HWM, promotion to a table lock will be attempted
- **PCT** (Percentage of Table; must be 1<=PCT<=100; default=100) - when the number of page locks on a table is between LWM and HWM, promotion to a table lock will be attempted only when the number of locks exceeds *(PCT*number-of-pages-in-table)/100*

Lock promotion settings are stored in the **sysattributes** table.

sp_dropglockpromote *{**"database"** | **"table"**}, objname*
Removes page lock promotion thresholds for a specific table or database.

sp_setpglockpromote *{**"database"** | **"table"**}, objname, lwm, hwm, pct*
sp_setpglockpromote ***"server"****, NULL, lwm, hwm, pct*
Defines or modifies the page lock promotion thresholds for a specific database or table (first form), or for the whole server (second form). When no new value should be set for LWM, HWM or PCT, specify NULL for the corresponding parameter.

sp_configure {**"page lock promotion LWM"** | **"page lock promotion HWM"** | **"page lock promotion PCT"**}, *value* (dynamic)
Equivalent to settings by **sp_setpglockpromote** *"server", NULL, lwm, hwm, pct*

sp_droprowlockpromote *{**"database"** | **"table"**}, objname*
sp_setrowlockpromote *{**"database"** | **"table"**}, objname, lwm, hwm, pct*
sp_setrowlockpromote ***"server"****, NULL, lwm, hwm, pct*
sp_configure {**"row lock promotion LWM" | "row lock promotion HWM"** | **"row lock promotion PCT"**}, *value* (dynamic)
Defines/modifies/drops row lock promotion thresholds for DOL tables. Otherwise identical to the above APL-table page lock promotion commands.

sp_sysmon syntax

sp_sysmon ***"begin_sample"***
sp_sysmon *{ **"end_sample"** | "hh:mm:ss" } [, section [, applmon]]*
The simplest way to invoke **sp_sysmon** is to specify a sample interval using *"hh:mm:ss"*. Using both the ***"begin_sample"*** and ***"end_sample"*** calls, sampling can also cover specific actions or statements. Warning: only a single **sp_sysmon** session should be run at a time, as multiple concurrent invocations will produce invalid results due to interference.
As the default **sp_sysmon** output can be voluminous, it can be limited to a specific section by specifying the *section* parameter as follows (in sp_sysmon output order):

***section* parameter**	**Resulting sp_sysmon output**
kernel	Kernel Utilization
wpm	Worker Process Management
parallel	Parallel Query Management
taskmgmt	Task Management
appmgmt	Application Management
esp	ESP Management
housekeeper	Housekeeper Task Activity
monaccess	Monitor Access to Executing SQL
xactsum	Transaction Profile
xactmgmt	Transaction Management
indexmgmt	Index Management
mdcache	Metadata Cache Management
locks	Lock Management
dcache	Data Cache Management
pcache	Procedure Cache Management
memory	Memory Management *(note: this is not very useful …)*
recovery	Recovery Management
diskio	Disk I/O Management

netio	Network I/O Management

The *applmon* parameter is valid only when specifying either "**appmgmt"** or **NULL** as the *section* parameter (it is ignored otherwise) :

appl_only	CPU, I/O, priority changes, and resource limit violations by application name.
appl_and_login	ibid., by application name and login name.
no_appl	Skips the application and login section (=default)

(ASE 11.0.x only) **sp_sysmon** *{ 1 | 2 | 3 … | 10 }* - In ASE 11.0.x, the only parameter is the duration (in minutes) of the sample interval, ranging from 1 to 10.

Update statistics

The **update statistics** commands will run at isolation level 0 for DOL tables; for APL tables, at level 1. In the statements below, *table_name* can optionally be qualified with the database name and/or owner name.
Table and index statistics, as stored in **systabstats** and **sysstatistics**, can be viewed and **sysstatistics** can be edited using the **optdiag** utility. To find the date/time when an **update statistics** command was last run for a table, run optdiag, or use this query:

select moddate from sysstatistics where id = object_id("*table_name*")

Except for **update partition statistics**, all **update statistics** commands can optionally specify the clause "***using*** *nr_steps* ***values"*** to force the specified number of steps for the histogram (default=20), as well as the ***"with consumers*** **=** *nr_of_consumers****"*** clause to force parallellism (intended for specific situations).

update statistics *table_name [index_name]*
For *table_name,* creates/updates a histogram for the leading index column in all indexes (when specified, only for *index_name*), and density statistics for all prefix subsets of the index columns.

update statistics *table_name [(column_list)]*
For *table_name,* creates/updates a histogram for the first column in the list and density statistics for all prefix subsets of the specified columns.

update index statistics *table_name [index_name]*
For *table_name,* creates/updates a histogram for all indexed columns in all indexes (when specified, only for *index_name*), c, and density statistics for all prefix subsets of the index columns.

update all statistics *table_name*
Performs **update index statistics** for all indexes, and creates/updates a histogram for all non-indexed columns; for partitioned tables, also performs **update partition statistics**.

update partition statistics *table_name [partition_nr]*
For partitioned tables, updates partition statistics for the specified, or all, partitions.

delete statistics *table_name [(column_list)]*
Deletes all statistics (both actual and simulated) for *table_name* from **sysstatistics**. With *column_list*, deletes only the statistics for those columns.

delete shared statistics
Deletes simulated statistics for configuration settings from **master..sysstatistics**.

Resource limits

sp_configure "allow resource limits", *0 | 1* (static)
Enables (1) or disables (0) resource limits being enforced.

sp_add_time_range *timerange_name, startday, endday, starttime, endtime*
Creates a new time range, running from *starttime* to *endtime* on each of the days between *startday* and *endday* (inclusive). Times are specified as "hh:mm"; to match an entire day, specify "00:00" as both start and end time. A default time range named **"at all times"** covering 24*7, already exists.
Example: **sp_add_time_range "9_to_5", Monday, Friday, "09:00", "17:00"**
To display existing time ranges: **select * from master..systimeranges**

sp_drop_time_range *timerange_name*
Drops the specified time range.

sp_modify_time_range *timerange_name, startday, endday, starttime, endtime*
Modifies the specified time range. See **sp_add_time_range** for parameter details.

sp_add_resource_limit *login_name, program_name, timerange_name, limit_type, limit_value [, enforced_code [, action_code [, scope_code]]]*
Creates a new resource limit, which applies to the specified *login_name* (when NULL, applies to all logins), in combination with running *program_name* (when NULL, applies to all programs), during the specified *timerange_name*. Possible values for *limit_type*:

- ***"row_count"*** - *limit_value* is the maximum number of rows that a query can return to the client; *enforced_code* must be 2 (=during execution) or omitted.
- ***"elapsed_time"*** - *limit_value* is the maximum number of elapsed seconds allowed for a transaction or query batch (note: this limit does not apply to individual queries!); *enforced_code* must be 2 (=during execution) or omitted.
- ***"io_cost"*** - when *enforced_code* is 1, *limit_value* is the optimizer's estimated I/O cost of a query (before execution); when *enforced_code* is 2, this is the query's actual I/O cost (during execution); when *enforced_code* is 3 (=default), it applies to both.

action_code is a number specifying what happens when exceeding the limit: 1: warning is issued; 2: batch is aborted (=default); 3: transaction is aborted; 4: session is killed. *scope_code* is a number specifying what the limit applies to: 1: applies to a query (only for ***"row_count"*** or ***"io_cost"***); 2,4,6: applies to a query batch, a transaction, or both, respectively (only for ***"elapsed_time"***) (defaults are 1 and 6).

sp_drop_resource_limit *login_name, program_name [, timerange_name, limit_type, enforced_code, action_code, scope_code]*
Drops one or more resource limits for the specified *login_name* in combination with running *program_name*, optionally combined with the other parameters. Specifying NULL for any parameter matches all possible values. See **sp_add_resource_limit** for parameter details.

sp_modify_resource_limit *login_name, program_name, timerange_name, limit_type, limit_value, NULL, action_code, NULL*
Modifies *limit_value* or *action_code* for an existing resource limit, identified by the other parameters. See **sp_add_resource_limit** for parameter details.

sp_help_resource_limit *[login_name [, program_name [, limit_time [, limit_weekday [, scope_code [, action_code]]]]]]*
Displays existing resource limits; optionally those existing at the specified time ("hh:mm") and/or weekday (e.g. "Sunday"). Specifying NULL for any parameter matches all possible values. See **sp_add_resource_limit** for parameter details.

Reorg

Note: **reorg** is available for DOL tables only.

reorg forwarded_rows *table_name [* ***with { resume, time =*** *minutes}]*
Moves forwarded rows back to their home page.

reorg reclaim_space *table_name [index_name] [* ***with {resume, time =*** *minutes}]*
Reclaims unused space from rows that were deleted or shortened in an update.

reorg compact *table_name [* ***with { resume, time =*** *minutes}]*
Performs both the **reclaim_space** and **forwarded_rows** actions.

reorg rebuild *table_name*
Defragments the table and its indexes; this requires sufficient free space and blocks all access to the table (Ex-Table lock). **"select into/bulkcopy/pllsort"** must be enabled; it is recommended to make a database dump afterwards.

(12.0) **reorg rebuild** *table_name index_name*
Rebuilds the specified index while the table & index remain accessible (Ex-Int lock).

(12.0) Java in the server

Java class definitions are stored in the system table **sysxtypes**; if retained, JARs are stored in **sysjars**.

sp_helpjava *[* ***"class"****, [class_name [,* ***"detail"*** *]] |* ***"jar"*** *[, jar_name]*
Displays information about existing classes and/or JARs. For classes, specify **"detail"** to display class details.

sp_configure "enable java", *0 | 1*
Enables (1) or disables (0) Java-in-ASE (static config option). This requires the ASE_JAVA option to be licensed.

Installing a Java class (named MyClass) into ASE:
1. Outside ASE, create an uncompressed .jar file with the following commands (assuming a standard JDK environment):

- **javac MyClass.java** (➔ produces file MyClass.class)
- **jar [-]cf0 MyJar.jar MyClass.class** (➔ produces MyJar.jar)

2. From the client system, install the .jar file into ASE (note: on NT, use "instjava" instead of "installjava") :
   - **installjava -f MyJar.jar** [**-update** | **-new**] [**-j** *jar_name*] **-S***servername* **-U***loginname* **-P***password* **-D***database_name*

   Default is **-new**; by default, the JAR is not retained; to retain it, use the **-j** option. Note that, in 12.0, classes are local to a database (i.e. not server-wide).

Removing a Java class (only possible from within ASE):
remove java *{* ***class*** *class_name [, class_name …]* | ***package*** *pckg_name [, pckg_name …]* | ***jar*** *jar_name [, jar_name]... [****retain classes****]* }
Drops the specified classes, packages or JARs from the current database. For JARs, the related classes are dropped as well, unless ***retain classes*** is specified.

Extracting a Java class:
From the client system, extract a JAR (which must have been retained), including all its Java classes, into a file (note: on NT, use "extrjava" instead of "extractjava") :
- **extractjava -j***jar_name* **-f***target_file_name* **-S***servername* **-U***loginname* **-P***password* **-D***database_name*

XP Server

XP Server can be used to perform actions outside ASE (such as executing OS commands or running other programs) from T-SQL code executed inside ASE. This is performed by the XP Server process, which is started and stopped by ASE when needed; therefore, do not start or stop XP Server explicitly yourself. Communicating with XP Server is done through so-called "Extended stored procedures" (ESPs), whose names start with "**xp**_".
The following steps are required to configure XP Server:
1. The **interfaces** file (NT: **SQL.INI** file) must contain the XP Server name with a unique port nr. XP server must run on the same system as the ASE server.
2. Add the XP Server to the ASE server using the command **sp_addserver PROD_XP** (for an ASE server named **PROD**; the _**XP** suffix is mandatory).
3. Choose the setting for config parameter **"xp_cmdshell context"** (see below).
4. Test the configuration by running a command like.

xp_cmdshell "*external_command***"** *[,* ***no_output*** *]*
Executes the specified command (max. 255 characters) on OS-level on the system where ASE (actually, XP Server) is running. The output from this command appears as a result set at the client. With **no_output** any output from this command is suppressed. Example: **xp_cmdshell "pwd"** (Unix) or **xp_cmdshell "dir"** (NT).

xp_deletemail, **xp_enumgroups**, **xp_logevent**, **xp_findnextmsg**, **xp_readmail**, **xp_sendmail**, **xp_startmail**, **xp_stopmail**
These extended stored procedures (available on Windows NT only) perform special functions; see the *ASE Reference Manual* for details.

sp_configure "xp_cmdshell context", *{* ***0*** | ***1*** *}* (dynamic)
Defines security settings for executing commands through **xp_cmdshell**. When set to 1 (=default), these execute under the OS account with the same name as the ASE login executing **xp_cmdshell**; if this account doesn't exist, **xp_cmdshell** fails. When set to 0, these commands execute under the OS account running the ASE server.

sp_addextendedproc *xp_proc_name, dll_file_name*
Only in the master database, creates an extended stored procedure *xp_proc_name*, which must be an existing entry point in the DLL file *dll_file_name*.

create proc[**edure**] *xp_proc_name* ***as external name*** *dll_file_name*
As **sp_addextendedproc**, but works in any database.

sp_dropextendedproc *xp_proc_name*
Drops the extended stored procedure *xp_proc_name* from the current database.

sp_freedll *dll_file_name*
Unloads a DLL used by XP Server for executing extended stored procedures.

sp_helpextendedproc *[xp_proc_name]*
Displays information about all extended stored procedures in the current database. With *xp_proc_name*, displays only that procedure.

Configuring shared memory dumps

Warning: ***this undocumented feature, which dumps the ASE shared memory contents to a file, is for special troubleshooting purposes only. Only Sybase Technical Support can read the contents of the resulting file; it contains no useful information for a DBA. Use this feature only when instructed by Sybase Technical Support or when regularly encountering serious error conditions. Avoid using this feature routinely or with a high frequency.***

sp_shmdumpconfig *[action, type, value, max_nr_dumps, directory, file_name, option_1, option_2, option_3]*
Displays or modifies configured dump conditions. A dump condition is a specific event, like a certain error. When this condition occurs, the server will dump the shared memory to a file which Sybase Technical Support can use for troubleshooting purposes. Note: use **sp_shmdumpsize** to check whether there's a risk of exceeding the maximum file size on your platform (like 2Gb). *action* can be:
- **"add"**, **"drop"**, **"update"** - adds/drops/changes a dump condition, respectively
- **"reset"** - resets the counter (counting the number of times a dump has been made) for a dump condition
- **"display"** - displays currently configured dump conditions

The actual dump condition is specified by a *type* and (usually) a *value*. *type* can be:
- **"error"** - *value* is an ASE error number (example: 605)
- **"module"** - *value* is the first of a block of 100 ASE error numbers (example: 1100; this corresponds to error numbers 1100-1199)
- **"signal"** - *value* is an OS signal number (example: 11)
- **"severity"** - *value* is a severity level (example: 17); the dump condition applies to errors with a severity >= *value*.
- **"timeslice"** - a timeslice error (*value* = **NULL**)
- **"panic"** - a server "panic" (which crashes the server) (*value* = **NULL**)
- **"defaults"** - updates default settings (*value* = **NULL**)

option_1, option_2 and *option_3* - These options determine which sections are included in the memory dump. Possible values are **"include_page"**, **"omit_page"**, **"default_page"**, **"include_proc"**, **"omit_proc"**, **"default_proc"**, **"include_unused"**, **"omit_ unused "**, **"default_unused"**. Legend: **"include_…"** or **"omit_…"** means a section is included or omitted; **"default_…"** means the default setting is used. Possible sections are **"…_page"** (data cache), **"…_proc"** (procedure cache), or **"…_unused"** (unused memory).

max_nr_dumps is the maximum number of times a memory dump will be performed for this dump condition. The count is reset when the server starts.

Examples: **sp_shmdumpconfig "add", "timeslice"**
sp_shmdumpconfig "add", "error", 691, 1, NULL, "err691.mdump"

sp_shmdumpsize *data_cache_option, proc_cache_option, size_in_Mb* ***output***
Calculates the expected size (in Mb) of the shared memory dump file for this ASE server; this value is returned as an output parameter of datatype **int**. *data_cache_option* and *proc_cache_option* specify whether the size of the data cache and procedure cache should be included. To exclude, specify **"Omit"**; to use the current default setting as configured though **sp_shmdumpconfig**, specify **"Default"**; to include, specify any other string (note: **"Omit"** and **"Default"** are case-sensitive).

sp_configure "dump on conditions", *{* ***0*** | ***1*** *}* (dynamic)
Enables (**1**) or disables (**0**, =default) shared memory dumps.

sp_configure "maximum dump conditions", *max_nr_conditions* (static)
Defines the maximum number of dump conditions that can be configured at one time through **sp_shmdumpconfig**. By default, this value is **10**.

Miscellaneous T-SQL issues

Outer join syntax
In an outer join, the table at the side of the "*" will have its qualifying rows returned in the result set, also when there is no matching row in the inner table. Example:

```
select T1.a, T2.b from T1, T2
where T1.key *= T2.key
```

In this example, all rows from table T1 will be included, even when there is no matching row in table T2. When this is the case, a NULL value will be substituted for T2.b in the result set.

"print" statement placeholders

Placeholders in a **print** (as well as in a **raiserror**) statement are in the form **%n!** (**n**=1..20). Errors in the placeholders result in a run-time (not a compile-time) error. Example: **print "My name is %1!, I'm %2! years old.", @my_name, @my_age**

The not-so-empty string
In an expression, an empty string (i.e., "") evaluates to a single space. Examples:
select "a" + "" + "b" results in **"a b"** (3 characters)
select "a" + NULL + "b" results in **"ab"** (2 characters)
select "a" + space(0) + "b" results in **"ab"** (2 characters)

exec @string_variable
Though not formally documented or supported, a stored procedure can be executed by specifying its name in a (var)char variable. Examples:
declare @string_var varchar(100) select @string_var = "sp_help"
exec @string_var "my_table"

select @string_var = "SYB_BACKUP…sp_who"
exec @string_var

Avoiding the "use <*database*>**" command**
System Stored Procedures can access other databases without actually changing the current database though the **use** command. Example: to display the layout of table **other_db..t1**, regardless of the current database: **exec other_db..sp_help t1**

Left-padding a number with zeroes
To convert a number to an n-character string with leading zeroes:
select right(replicate("0", n) + convert(varchar, 123),n)
this returns **"000123"** for **n** = 6

Displaying reserved words
To display all reserved words in your current ASE version, run this query:
select name from master..spt_values where type = "W"

CASE expression
The CASE-expression can have the following 4 forms:
- **case when** *x=y* **then** *expr* […] **else** *expr* **end**
- **case** *expr* **when** *expr* **then** *expr* [...] **else** *expr* **end**

Note: when the **else**-clause is omitted, **else NULL** is used.
- **coalesce** *(val1, val2, val3…, valn)* : returns first non-NULL value
- **nullif** *(x,y)* : when x=y, returns NULL, otherwise returns x

Point-characteristic functions
In ASE pre-11.5, the CASE expression doesn't exist, but similar functionality can be implemented through point-characteristic functions. The following example returns "Nine" when @n=9, "Ten" when @n=10, and "What?" otherwise:
select right("What?" + substring("Nine ", 1-abs(sign(9-@n)), 5) +
substring("Ten ", 1-abs(sign(10-@n)), 5) ,5)

Miscellaneous DBA issues

The magic number in master..sysdevices
Logical page numbers on a database device are based on the *vdevno* (which is specified in "disk init") and the magic number 16777216 (= 2^{24}): the first page on a device has page number **vdevno*16777216**. To find the vdevno's currently in use:
select low/16777216 from master..sysdevices
where status & 2 = 2 order by 1

Displaying the contents of a dump tape
To display the contents of a dump tape (it's safe: it doesn't really load the dump !):
load tran tempdb from *tape_device* **with listonly [= full]**

Issues with BCP-in
To BCP data into a table, "fast" mode (=minimally logged) will be used when all of the following are true:
- the table has no indexes
- the table has no triggers; in 12.0, triggers may exist must have been disabled using **alter table** *table_name* **disable trigger**
- the "select into/bulkcopy/pllsort" database option is enabled

In all other cases, "slow" mode (=fully logged) will be used.
When BCP-ing into a table in either mode, rules, triggers and RI constraints are not executed, but defaults are. This can lead to problems when the table has first been BCP'd out and then BCP'd back in: for columns with a default, rows originally having NULL values in such columns will not be NULL anymore after being BCP'd back in,

but will have the default value instead (solution: drop the default before BCP-in and restore it afterwards).

Starting/stopping servers as WindowsNT services
On WindowsNT, Sybase servers are installed as NT services. The name of the NT service is composed of the servertype, followed by the servername:

Server type	Name of NT service
ASE (aka. SQL Server)	SYBSQL_<servername>
Backup Server	SYBBCK_<servername>
Monitor Server	SYBMON_<servername>
Replication Server	SYBREP_<servername>
Rep.Manager Server	SYBRSM_<servername>
LTM	SYBLTM_<servername>

To start/stop an ASE server named "PROD" on the local NT host, using the following DOS command: **net [start | stop] SYBSQL_PROD**

To start/stop a backup server named "PROD_BS" on a remote NT host named "\\otherhost", use any of the commands **sc** or **netsvc** (part of the NT resource kit):
sc \\otherhost [start | stop] SYBBCK_PROD_BS
netsvc SYBBCK_PROD_BS \\otherhost [start | stop]

The parameters used when a server is started as a service, are in the registry key HKEY_LOCAL_MACHINE\SOFTWARE\SYBASE\Server\<*servername*>\Parameters.

Minimally logged operations
The following operations are "minimally logged" (a.k.a. unlogged), meaning that no actual row modifications are written to the transaction log, but only page (de)allocations:
- **select…into**
- fast **bcp**-in (i.e. for tables without indexes or triggers)
- **writetext** (unless the "with log" option is used)
- **create index** (when executed in parallel)
- **reorg rebuild** *table_name*
- ***(12.0)*** **alter table***…{**add|modify|drop**} column* (when data copying is required)
- **truncate table**
- **alter table … lock …** (when changing between APL and DOL schemes)
- **dump transaction … with truncate_only** (also: **"trunc log on chkpt"** option)
- **dump transaction … with no_log** (this command really logs nothing at all)

With the exception of **truncate table**, when any of these commands are executed, the transaction log of that database cannot be dumped anymore as it doesn't contain all data required to fully recover these operations. Therefore, a database dump should be performed first. Note: the commands marked "➢" above require the **"select into/bulkcopy/pllsort"** database option to be enabled.

List of essential DBA tasks

Here's a quick overview of the most essential (and some optional) DBA tasks that need to be performed regularly in any ASE environment, and their typical frequency. See the *ASE System Administration Guide* for details and additional guidelines.

Essential DBA Tasks	Frequency (typical)
Perform database and transaction log dumps	daily / hourly
Run DBCC checks (preferably **dbcc checkstorage**)	weekly
Run UPDATE STATISTICS on user tables	weekly
Regular stop and restart of ASE server	monthly (at least)
Monitor server errorlog for anomalies	daily
Troubleshoot unforeseen emergencies	ad-hoc (read: daily ;-)
Attend to (end)user/developer needs	ibid.

Additional / Optional DBA Tasks	Frequency (typical)
Monitor growth of data volume and log space usage	daily / weekly
Defragment tables (**reorg** or rebuild clustered index)	monthly / quarterly
Monitor/tune server resource usage with **sp_sysmon**	when possible
Set up the **dbccdb** database for **dbcc checkstorage**	once
Set up the **sybsyntax** database (not in ASE 12.0 !)	once

ASE environment variables

The following Unix environment variables are used by ASE (on NT, read %VARIABLE% instead of $VARIABLE):

Environment variable	Function
$SYBASE	Installation directory of the all Sybase software
$PATH	Should contain $SYBASE, and the *bin* subdirectories;

	on Unix, also the *install* subdirectory; on NT, also the *dll* subdirectories.
$DSQUERY (optional)	Default server name for client utilities (like *isql*)
$DSLISTEN (optional)	Default server name when ASE is started
$LD_LIBRARY_PATH	Should contain the *lib* subdirectories (not on NT)
$LIB_PATH	(AIX only) Should contain the *lib* subdirectories
$SHLIB_PATH	(HP-UX only) Should contain the *lib* subdirectories
%LIB%	(NT only) Should contain the *lib* subdirectories
(12.0) $SYBASE_ASE	ASE installation directory (below $SYBASE)
(12.0) $SYBASE_OCS	OpenClient installation directory (below $SYBASE)
(12.0) $SYBASE_FTS	Full-text search installation directory (below $SYBASE)
(12.0) $SYBASE_SYSAM	License manager install. directory (below $SYBASE)

The following variables are only relevant when using Replication Server 12.0:

(12.0) $SYBASE_REP	RepServer installation directory (below $SYBASE)
(12.0) $SYBASE_RSM	RepServer Manager install. directory (below $SYBASE)
(12.0) $SYBASE_RSP	RepServer plug-in for Sybase Central installation directory (below $SYBASE)

Sybase resources on the web

Here's an arbitrary selection of some Sybase-related WWW links:

The Sybase FAQ: `http://www.isug.com/Sybase_FAQ/`
Michael Peppler's SybPerl: `http://www.mbay.net/~mpeppler`
Ed Barlow's System Stored Procs: `http://www.edbarlow.com`
"sqsh" (a great "isql" replacement): `http://www.sqsh.org`
ISUG (International Sybase User Group): `http://www.isug.com`
On-line Sybase manuals (including downloadable manuals in PDF format): `http://www.sybase.com/support/manuals/`
Sybase-related newsgroup on Usenet: `comp.databases.sybase`
More newsgroups exist at Sybase's own news server: `forums.sybase.com`

This document, as well as "sybinit4ever", the Replication Server Quickref Guide and other useful Sybase stuff can be found at `http://www.sypron.nl`

How to assemble the pages into a booklet:

1. Print the file (paper size should be A4). You may need to adjust the page margins or the space between the columns if it doesn't print correctly; there should be about 7 millimeters (~ 5/12 inch) free space at the left and right margins (if you can't get it printed correctly, you can change the lay-out to 1 column/page, and so some manual cut-and-pasting).

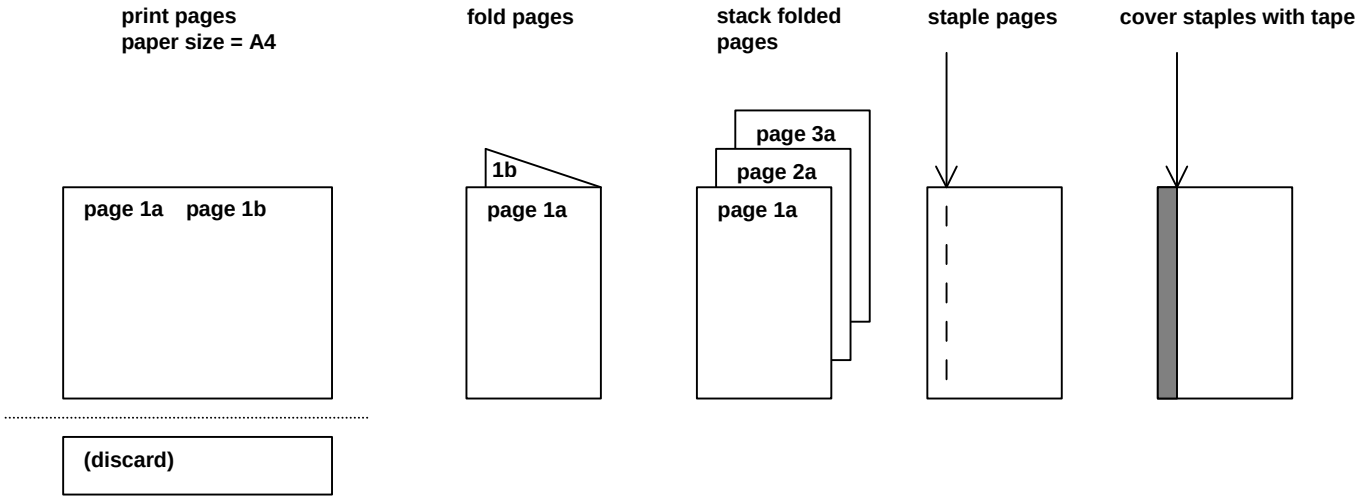
2. Cut off the blank bottom part of the pages

3. Fold the printed pages, so that there is one column on each folded side.

4. Stack the folded pages

5. Staple the folded pages together at the left-hand margin

6. If you don't like the sight of those staples, cover them with a bit of sticky tape.



Do you find this ASE Quick Reference Supplement useful ?

Then check out ***"The Complete Sybase ASE Quick Reference Guide"*** , the only truly complete ASE Quick Reference Guide that exists. This 96-page, fit-in-your-pocket book contains all commands, functions, procedures and other information you need most often when working with ASE 11.9, 12.0 or 12.5 It includes full command syntax, command descriptions and examples, as well as a selection of useful undocumented commands. There's also a handy 10-page index. It's probably the most useful book any ASE DBA or developer could want to have.

More information, including a preview of some of the actual pages of the book, is available at www.sypron.nl/qr . Online orders are accepted (using a secure payment interface).