New Features Guide

Adaptive Server® Enterprise 16.0

DOCUMENT ID: DC00641-01-1600-01
LAST REVISED: March 2014

Contents

Overview of Version 16.0

SAP® Adaptive Server® Enterprise version 16.0 is a major release and includes key enhancements in scale-up and performance, managing large data sizes, increased data availability, security and auditing, and ease of management and maintenance.

- Scale-up and performance improvements include optimized buffer management to minimize contention.
- Metadata and latch management improvements allow for high throughput
- Increased data availability from partition level locking enable concurrent DDL and DML on a table
- Improvements to large data management build on those in SAP® ASE version15.7, and include relaxed query limits, query plan optimization with star joins, and related query performance enhancements including `SORT` operator performance improvements.
- New ways to compress data, including index compression, combined with page- and column-level compression, as well as other compression techniques added in earlier releases.
- Enhancements to SCC include monitoring and management improvements, enhancements to threshold-based events, query plan and execution statistics in HTML format, and advisors that can help you make more informed decisions about your data, specifically when you are using compression.

Feature	Description
Scalability and Performance Features	
Linear scale-up	Supports linear scale-up to 64 cores
Dynamic thread support	Execute parallel query plans faster and with fewer resources
Index compression	More efficient data storage, reduced memory consumption, and improved performance due to lower I/O demands.
Support for multiple triggers	Create multiple triggers, as well as specify the order in which triggers fire after statement execution
Partition locks	Increases data availability by providing locking to a finer granularity, which allows you to access additional partitions for concurrent DDL and DML statements.
Configuration History Tracking	Use the **sp_confighistory** system procedure to track changes to the server configuration

Feature	Description
Threshold-based event monitoring table	Configure, record, and list threshold events
Extend execution time monitoring	The `monThresholdEvent` monitoring table includes one row for each event recorded by SAP ASE.
monCachedStatement updates	Updates the metrics for selected columns approximately every 5 seconds for completed and in-process queries
Installer enhancements	Specify the user for the SAP ASE process
Raise database limits	Increases some column and table restrictions
create or replace commands	Create a new object, or replace an existing object with the same name.
Security and Auditing Features	
Encrypt entire database	Encrypt the full database.
Remove Residual Data	Mark data as sensitive, and configure SAP ASE to erase residual data from disk after delete or update operations.
Full text audit	Full-text audit information for these commands: • **select** • **insert** • **delete** • **update** • **select into**
Interoperability with HANA	
CIS support for HANA	Adds the native ODBC interface to the Component Integration Services (CIS) so you can connect directly to a HANA server from SAP ASE
SCC Features	
Schedule database and transaction backups	Schedule database and transaction backups
Compression advisor	Identifies tables that can benefit from compression, and obtains compression estimates and recommendations.
System configuration limit alerts	Configure alerts to fire when certain resources exceed a configured threshold.
Scalability	Manage up to 250 servers.

Feature	Description
Tool Support	
sybrestore enhancements	Restore an SAP ASE server after a master database corruption. New parameters for interactive and noninteractive modes.

Increased Data Availability with Partition Locking

The partition-level locking feature increases data availability by creating finer locking granularity, which allows access to other partitions for concurrent DDL and DML statements.

Partition Locks

Partition locks are an attribute of a partitioned table and are applicable to all types of partitioning.

- Intent partition lock – indicates that page-level or row-level locks are held on a partition of table. SAP ASE applies an intent partition lock with each shared or exclusive partition lock, so an intent lock can be either an exclusive lock or a shared lock. Setting an intent lock prevents other transactions from acquiring conflicting partition-level locks on the table as the partition locks are in effect for the transaction.
- Shared partition lock – similar to a shared page or row lock, except that it affects the entire partition. For example, SAP ASE applies a shared partition lock for a **select** command with a **holdlock** clause if the command does not use an index. A **create nonclustered index** command also acquires a shared partition lock.
- Exclusive partition lock – similar to an exclusive page or row lock, except that it affects the entire partition. For example, SAP ASE applies an exclusive partition lock during a **create clustered index** command. **update** and **delete** statements on data-only-locked partitions require exclusive partition locks if their search arguments do not reference indexed columns of the object.
- Covering partition lock – these locks are required when DDLs and DMLs do not know the *partitionid* while acquiring the fine-grained lock.

For information about other types of locks, see *Performance and Tuning Series: Locking and Concurrency Control*.

This example table shows the respective partition locks of page or row locks SAP ASE uses for basic SQL statements. For these examples, there is an index on `acct_number`.

Statement	Allpages-Locked Table	Datarows-Locked Table
`select balance from account where acct_number = 25`	Intent shared partition lock Shared page lock	Intent shared partition lock Shared row lock

Statement	Allpages-Locked Table	Datarows-Locked Table
`insert account values (34, 500)`	Intent exclusive partition lock Exclusive page lock on data page Exclusive page lock on leaf index pages	Intent exclusive partition lock Exclusive row lock
`delete account where acct_number = 25`	Intent exclusive partition lock Update page locks followed by exclusive page locks on data pages and leaf-level index pages	Intent exclusive partition lock Update row locks followed by exclusive row locks on data rows
`update account set balance = 0 where acct_number = 25`	Intent exclusive partition lock Update page locks followed by exclusive page locks on data pages and leaf-level index pages	Intent exclusive partition lock Update row locks followed by exclusive row locks on data rows.

Partition Lock Name

SAP ASE uniquely identifies the partition lock with a combination of database ID (*dbid*), object ID (*objid*), and partition ID (*ptnid*).

Unknown Partition Locks

For unknown partition locks, SAP ASE uses a special partition lock with a partition ID value of '-1'. It indicates a partition lock on all of the table partitions. In specific cases, SAP ASE may not know the partition ID of the data row and would acquire a partition lock with partition ID '-1'.

Enabling Partition Locking

Use **sp_chgattribute** to enable or disable partition-level locking. By default, partition locking is disabled.

Partition-level locking can be enabled only on user tables with more than one data partition. Partition-level locking can not be enabled for system tables and temporary tables.

The syntax is:

```
sp_chgattribute  objectname, 'ptn_locking', value
```

Parameters:

objectname – is the name of the table on which to change **ptn_locking**.

`value` – set to 1 to enable and 0 to disable partition-level locking.

Permissions:

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

Examples

This example enables partition-level locking for the *authors* table:

```
sp_chgattribute authors, "ptn_locking", 1
```

This example disables partition-level locking for the *authors* table:

```
sp_chgattribute authors, "ptn_locking", 0
```

View Partition Locks with sp_lock

Use **sp_lock** to view partition locks currently held by SAP ASE.

View locked partitions in the *partitionid* column of the **sp_lock** report.

This example displays locks, including partition locks, currently held by SAP ASE.

```
sp_lock
go

fid spid loid locktype table_id partitionid page row dbname class
context
--- ---- ---- -------- -------- ----------- ---- --- ------ ------
-------
0 13 26 Ex_intent 420193516 0 0 0 master Non Cursor Lock
0 13 26 Ex_intent_partition  420193516 452193630 0 0 master Non
Cursor Lock
0 13 26 Ex_page 420193516  452193630 4993 0 master Non Cursor Lock
0 14 28 Ex_intent 420193516 0 0 0  master Non Cursor Lock
0 14 28 Ex_intent_partition 420193516 468193687 0 0  master Non
Cursor Lock
0 14 28 Ex_page 420193516 468193687 5001 0 master Non Cursor Lock
0 16 32 Sh_intent 1006623598 0 0 0 master Non Cursor Lock
```

See also

- *sp_lock* on page 164

View Partition Locks with sp_familylock

Use **sp_familylock** to display the partition locks held by a family.

View locked partitions in the *partitionid* column of the **sp_familylock** report.

This example displays the partition locks currently held by a family. The class column displays the cursor name for locks associated with a cursor for the current user and the cursor id for other users.

```
sp_familylock
go

Fid spid loid locktype table_id partitionid page dbname class context
------ ------ ------ ------------------- ---------------
---------------- -------- ----------- ----------
------------------------------

25 19 50 Sh_cpartition 672002394 -1 0 userdb Non Cursor Lock LOCK
```

```
CONTEXT VALUES
25 19 50 Sh_partition 672002394 688002451 0 userdb Non Cursor Lock
LOCK CONTEXT VALUES
25 20 50 Sh_cpartition 672002394 -1 0 userdb Non Cursor Lock LOCK
CONTEXT VALUES
25 20 50 Sh_intent_partition 672002394 688002451 0 userdb Non Cursor
Lock LOCK CONTEXT VALUES
25 20 50 Sh_partition 672002394 704002508 0 userdb Non Cursor Lock
LOCK CONTEXT VALUES
25 25 50 Sh_intent 672002394 0 0 userdb Non Cursor Lock LOCK CONTEXT
VALUES

(6 rows affected)
(return status = 0)
```

See also

- *sp_familylock* on page 162

View Partition Locks with monLocks and monDeadLock

View locked partitions in the *partitionid* column of the `monLocks` and `monDeadLock` monitoring tables.

monLocks

Description	Datatype	Attribute	Description
`partitionid`	int	Null	Unique identifier for the partition.

monDeadLock

Description	Datatype	Attribute	Description
`partitionid`	int	Null	Unique identifier for the partition.

See *Performance and Tuning Series: Monitoring Tables*.

Deadlocks and Lock Timeouts

Deadlock detection and lock timeouts for a partition lock are performed in the same way as for table locks.

See *Performance and Tuning Series: Locking and Concurrency Control*.

Partition Lock Promotion

Table lock promotion promotes locks from fine-grained locks to table locks. Partition lock promotion promotes locks from fine-grained locks to partition locks.

For partition lock promotion to occur, set the partition lock promotion threshold to a nonzero value.

Partition locks use the same semantics as table locks. Page or row locks belonging to a partition under a single partition scan that exceeds its lock promotion threshold can then trigger lock promotion to a partition lock.

When a page or row lock is acquired with an unknown partition, lock promotion to the partition lock is completely disabled. In these situations, locks can only be promoted to table level.

SAP ASE supports two different types of lock promotion:

- Row or page to table lock – if a task acquires locks within a single scan of a table on as many number of rows or pages that exceed the table-level lock promotion threshold, then SAP ASE tries to acquire shared table or exclusive table locks on the corresponding table and replace all existing rows or pages in that table. Table-level lock promotion is also triggered when row or page locks are acquired with covering partition when the owning partition is not known for the lock promotion.
- Row or page to partition lock – during partition lock promotion the appropriate shared or exclusive partition lock is acquired and all of the fine-grained locks acquired as part of the scan (or DML) and belonging to the partition are released.
  - Partition lock promotion promotes shared fine-grained locks to shared partition locks.
  - Partition lock promotion promotes exclusive fine-grained locks to exclusive partition locks.

Setting Partition Lock Promotion Thresholds

The **sp_setpglockpromote_ptn** and **sp_setrowlockpromote_ptn** system procedures set partition-lock promotion thresholds at the server, database, and table level.

See also

- *sp_setpglockpromote_ptn* on page 175
- *sp_setrowlockpromote_ptn* on page 176

Dropping Partition Lock Promotion Thresholds

The **sp_ dropglockpromote_ptn** and **sp_droprowlockpromote_ptn** system procedures remove partition-lock promotion thresholds at the server, database, and table level.

See also

- *sp_ dropglockpromote_ptn* on page 167
- *sp_droprowlockpromote_ptn* on page 168

Lock Compatibility and Lock Sufficiency

Lock compatibility and lock sufficiency are two basic concepts that support issues of locking and concurrency.

- Lock compatibility – if a task holds a lock on any resource (for example, row, page, partition, or table) can another task also hold a lock on the same resource?
- Lock sufficiency, for the current task – is the current lock held on any resource (for example, row, page, partition, or table) sufficient if the task needs to access the resource again?

See *Performance and Tuning Series: Locking and Concurrency Control > Introduction to Locking*.

Partition Lock Compatibility

A summarized table about partition lock compatibility, showing when locks can be acquired immediately.

	Can Another Process Acquire:					
Lock Type Held:	**An Exclusive Partition Lock?**	**A Shared Partition Lock?**	**An Exclusive Intent Partition Lock?**	**A Shared Intent Partition Lock?**	**An Exclusive Covering Partition Lock?**	**A Shared Covering Partition Lock?**
An exclusive partition lock	No	No	No	No	No	No
A shared partition lock	No	Yes	No	Yes	No	Yes
An exclusive intent partition lock	No	No	Yes	Yes	No	No

	Can Another Process Acquire:					
Lock Type Held:	**An Exclusive Partition Lock?**	**A Shared Partition Lock?**	**An Exclusive Intent Partition Lock?**	**A Shared Intent Partition Lock?**	**An Exclusive Covering Partition Lock?**	**A Shared Covering Partition Lock?**
A shared intent partition lock	No	Yes	Yes	Yes	No	Yes
An exclusive covering partition lock	No	No	No	No	Yes	Yes
A shared covering partition lock	No	Yes	No	Yes	Yes	Yes

Partition Lock Sufficiency

A sufficiency matrix table for partition locks.

	Is That Lock Sufficient if the Task Needs:					
Lock Type Held:	**An Exclusive Partition Lock?**	**A Shared Partition Lock?**	**An Exclusive Intent Partition Lock?**	**A Shared Intent Partition Lock?**	**An Exclusive Covering Partition Lock?**	**A Shared Covering Partition Lock?**
An exclusive partition lock	Yes	Yes	Yes	Yes	Yes	Yes
A shared partition lock	No	Yes	No	Yes	No	Yes
An exclusive intent partition lock	No	No	Yes	Yes	No	No

	Is That Lock Sufficient if the Task Needs:					
Lock Type Held:	**An Exclusive Partition Lock?**	**A Shared Partition Lock?**	**An Exclusive Intent Partition Lock?**	**A Shared Intent Partition Lock?**	**An Exclusive Covering Partition Lock?**	**A Shared Covering Partition Lock?**
A shared intent partition lock	No	No	No	Yes	No	No
An exclusive covering partition lock	No	No	No	No	Yes	Yes
A shared covering partition lock	No	No	No	No	No	Yes

Schema Lock Compatibility

A summarized table about schema lock compatibility, showing when locks can be acquired immediately.

	Can Another Process Acquire:	
Lock Type Held:	**An Exclusive Schema Lock?**	**A Shared Schema Lock?**
An exclusive schema lock	No	No
A shared schema lock	No	Yes

Schema Lock Sufficiency

A sufficiency matrix table for schema locks.

	Is That Lock Sufficient if the Task Needs:	
Lock Type Held:	**An Exclusive Schema Lock?**	**A Shared Schema Lock?**
An exclusive schema lock	Yes	Yes

	Is That Lock Sufficient if the Task Needs:	
Lock Type Held:	**An Exclusive Schema Lock?**	**A Shared Schema Lock?**
A shared schema lock	No	Yes

Improved Concurrency for Partition-Level Online Operations

Certain partition-level operations can concurrently operate on different partitions of a table. DML can also concurrently operate on the table while the partition level online operation is running.

These include:

- **alter table … split partition**
- **alter table … merge partition**
- **alter table … move partition**
- **alter table … drop partition**
- **truncate partition**
- **dbcc checkindex**
- **dbcc checktable**
- **dbcc tablealloc**
- **dbcc indexalloc**

Partition-Level Online Operation Syntax

To allow for greater data availability, **alter table** and **truncate partition** commands include the **with online** sub-clause in the partition clause.

Partition locking must first be enabled.

The syntax for **alter table** is:

```
alter table table_name
{merge | drop | move | split} partition partition_name
existing_clause
with online
```

where:

table_name – is the name of the table to modify.

partition_name – specifies the name of the partition to merge, drop, move, or split.

existing_clause – is the partition subclause, depending on the partition action specified.

with online – executes in an online mode. Enables concurrent access to the table.

The syntax for **truncate partition** is:

```
truncate table table_name
   [partition partition_name]
   [with online]
```

where:

table_name – is the name of the table to truncate.

partition_name – specifies the name of the partition to truncate.

with online – executes in an online mode. Enables concurrent access to the table.

Concurrency with Partition-Level Online Operations

Multiple partition-level online operations can be executed concurrently on different partitions of a table.

All DMLs — that is, **select** (but not **select into**), **insert**, **update**, and **delete** — can operate on a table while partition-level online operations are in progress.

DMLs can operate on all the partitions other than the partitions being operated by partition-level online operations.

DMLs with appropriate use of predicates leads to SAP ASE making use of the partition elimination technique. This may lead to DMLs operating on only minimal required set of partitions.

Without partition elimination, DMLs operate on all partitions of table.

DMLs will be aborted when it operates on the partition that is concurrently being operated by partition-level operations.

Note: To avoid concurrent DMLs leading to errors, DMLs can be written using appropriate predicates that makes use of partition elimination technique. This results in DMLs operating on a minimal set of partitions.

For partition-level online operations, such as splitting a partition or moving a partition, the table must have a local unique index. The move command is allowed for concurrent DMLs to all partitions in the table, including ones being operated by split.

Partition-Level Online Operations with Global Index

Partition-level operations behavior for a table having a global index.

Concurrent DML access to a table does not use global index while a partition-level operation is in progress on the table. SAP ASE uses alternate local index scan or table scan for concurrent DML scans.

Multiple partition-level operations on different partitions can operate concurrently. Each one does not perform global index rebuild. Only the last committed partition-level operation performs global index rebuild.

The last partition-level operation leading to abort may result in global index marked as suspect. The global index in such case has to be explicitly rebuilt.

Schema Locks

Schema locks allow enhanced partition-level operations to update table schema or metadata by achieving isolation from concurrent operations.

The new schema locks are:

- Shared schema lock – indicates that a task is using the current schema of the table for query execution. Scans and DMLs acquire this lock before starting the execution of the query.
- Exclusive schema lock – indicates that a task is changing the schema of the table. The partition-level operation acquires this lock to update the schema of the table.

Component Integration Services Support for HANA Server

SAP ASE version 16.0 adds the native ODBC interface to Component Integration Services (CIS), which lets you connect directly to a HANA server from SAP ASE.

See the *Component Integration Services Users Guide* for information about using CIS to connect to remote servers.

Configuring CIS for HANA

You must install the SAP HANA client package, which includes the ODBC drivers, on the same machine that is running SAP ASE.

See *Client Installation and Update Guide* in your SAP HANA database documentation.

Creating SAP HANA as an ODBC Data Source on Windows

The ODBC Administrator provides a central place for creating and managing ODBC data sources on the Windows platform.

When using 32-bit ODBC drivers, use the 32-bit ODBC Administrator to manage data sources; for 64-bit clients, use the 64-bit ODBC Administrator.

1. From the Windows Start window, select **Settings > Control Panel > Administrative Tools > Data Sources (ODBC)**.
2. Click **Add**.
3. Select **HDBODBC** from the list of drivers.
4. Click **Finish**.
5. In the ODBC Configuration window, enter the Data Source Name and Server:Port . For example, to connect to a server named "HANA_DB" on host "hanaserver" with address 157.133.66.75 using TCP/IP protocol and port 1870, select TCP/IP and enter:

   ```
   host=hanaserver:1870
   ```

   To use a host network address, enter:

   ```
   host=157.133.66.75:1870
   ```
6. Click **Ok**.

Adding SAP HANA to the SAP ASE Interfaces File

You must add the HANA server entry to the SAP ASE interfaces file before you can connect to it.

Using vi or a similar text editor, add an entry for the SAP HANA server to the interfaces file or sql.ini file.

This example adds an entry on UNIX for the HANA_DB server using port number 1870 and the `libodbcHDB.so` SAP HANA ODBC driver:

```
HANA_DB
      query tcp ether 157.133.66.75 1870 libodbcHDB.so
```

This example adds an entry on Windows for the HANA_DB server using the HDBODBC (HDBODBC32 for Windows 32-bit) driver, which is registered in the Windows registry:

```
[HANA_DB]
Query=NLWNSCK,157.133.66.75,1870,HDBODBC
```

Note: Because PCI Bridge uses HDBODBC as the default value, you need not include an HDBODBC entry in the interfaces file (`sql.ini`) for Windows.

Configuring the PCI Bridge and PCA/ODBC

SAP ASE uses Pluggable Component Interface (PCI) Bridge, which implements on-demand software dispatching, to load shared objects when it invokes a target function.

ODBC Driver Manager is typically bootstrapped from the pluggable component adapter for ODBC (PCA/OBDC), which is configured with PCI Bridge. PCA/ODBC acts as a broker, managing service requests between the SAP ASE and the ODBC Driver Manager. PCA/ODBC forwards and controls requests in both directions—from the SAP ASE to ODBC Driver Manager, and vice versa.

ODBC Driver Manager component is independent of SAP ASE. You can change or upgrade these software components at any time without upgrading or rebuilding SAP ASE.

Note: You must enable PCI Bridge for PCA/ODBC before it can use ODBC Driver Manager. SAP ASE requires the `sybpcidb` database when you enable PCI Bridge. The `sybpcidb` contains all configuration data for PCI Bridge and its associated PCAs, such as PCA/ODBC.

1. As system administrator, create the `sybpcidb` database using the **srvbuild** or **sybconfig** installation and configuration utilities, or by running the `${SYBASE}/${SYBASE_ASE}/scripts/installpcidb` script (see the installation guide).
2. Enable PCI Bridge:

   ```
   sp_configure 'enable pci', 1
   ```

   See *Setting Configuration Parameters* in *System Administration Guide: Volume 1*.
3. Restart SAP ASE.

4. Use **sp_odbcconfig** to configure these PCA/ODBC parameters:
   - **pca_odbc_load_dir** – (UNIX only) sets the absolute path to the location for the ODBC drivers. The **pca_odbc_load_dir** makes an array of multiple locations. That is, if there are multiple ODBC drivers installed on the machine, **pca_odbc_load_dir** creates an array of them.

     The default value for **pca_odbc_load_dir** is `/usr/sap/hdbclient`.

     a. Change to the `sybpcidb` database:

     ```
     use sybpcidb
     ```

     b. Set the path for **pca_odbc_load_dir**:

     ```
     sp_odbcconfig 'add', 'pca_odbc_load_dir',
     'path_to_driver_location'
     ```

     c. Restart SAP ASE after making changes to **pca_odbc_load_dir**.

     To remove drivers that are no longer used:

     ```
     use sybpcidb
     go
     sp_odbcconfig 'delete', 'pca_odbc_load_dir',
     'path_to_driver_location'
     ```

     Note: On Windows, ODBC drivers are administered and maintained by the system registry during driver component installation. See your Windows platform documentation and specific driver vendor documentation.

   - **pca_odbc_root_dir** – UNIX systems store the ODBC data sources in the `odbc.ini` file. **pca_odbc_root_dir** indicates the path to the directory that contains `odbc.ini`. The default value for **pca_odbc_root_dir** is $SYBASE.

     The format for an entry in `odbc.ini` for the HANA_DB server is:

     ```
     [HANA_DB]
     DRIVER = libodbcHDB.so
     SERVERNODE = hanaserver:1870
     ```

     Note: On Windows, use the ODBC registry to configure the ODBC data sources. See your Windows documentation.

     On UNIX systems, use **sp_odbcconfig** to set the value for **pca_odbc_root_dir**:

     a. Change to the `sybpcidb` database:

     ```
     use sybpcidb
     ```

     b. Enter the new path:

     ```
     sp_odbcconfig 'update', 'pca_odbc_root_dir', 'old_path',
     'new_path'
     ```

     where *new_path* indicates a relative location to $SYBASE. Preface *new_path* with a forward slash to indicate an absolute location to the root directory.

     Note: You must restart SAP ASE if ODBC Driver is already loaded under control of PCA/ODBC. SAP ASE consults the DSN records stored under `odbc.ini` only after you enable **pca_odbc_dsn_enabled**.

- **pca_odbc_dsn_enabled** – enables or disables the DSN mode for the PCA/ODBC subsystem. Values are:
  - 0 – disables DSN (the default). SAP ASE automatically gathers DSN information from the SAP directory control layer (DCL) from the interfaces file.
  - 1 – enables DSN. SAP ASE consults the DSN records stored under `odbc.ini`.

5. (UNIX only) Add `$SYBASE/DataAccess64/ODBC/dm/lib64` to the LD_LIBRARY_PATH.

Adding Server Class HANAODBC

SAP ASE version 16.0 adds the HANAODBC server class, which enables you to connect to the SAP HANA server using ODBC.

You must install the HANA ODBC driver on the same machine that is running SAP ASE.

1. To add the HANAODBC server class, execute:

   ```
   sp_addserver logical_server_name, HANAODBC, ODBC_DSN_name
   ```

   This example adds `big_hana_server` as the `Big_company` DSN:

   ```
   sp_addserver big_hana_server, HANAODBC, Big_company
   ```

2. Use **sp_addexternlogin** to map the SAP ASE login account and password.

   This example configures the system administrator to use the name 'HANA_login' with "HANA_login_password" to log in to the remote server named "big_hana_server":

   ```
   sp_addexternlogin big_hana_server, sa, HANA_login,
   HANA_login_password
   ```

Datatype Mapping Between SAP ASE and HANA

When you create proxy tables, SAP ASE datatypes are mapped to corresponding HANA datatypes.

HANAODBC maps these SAP ASE datatypes to HANA datatypes when you use **create table** to create proxy tables:

SAP ASE Datatype	HANA Datatype
tinyint	tinyint
smallint	smallint
int	integer
integer	integer
bigint	bigint

SAP ASE Datatype	HANA Datatype
unsigned smallint	integer
unsigned int	bigint
unsigned bigint	Not supported
numeric (precision, scale)	decimal (precision, scale)
decimal (precision, scale)	decimal (precision, scale)
float(precision)	float(precision)
double (precision)	float
real	real
smallmoney	decimal (10,4)
money	decimal (19,4)
smalldatetime	timestamp
datetime	timestamp
date	date
time	time
char(n)	char(n)
varchar(n)	varchar(n)
unichar(n)	nchar(n)
univarchar(n)	nvarchar(n)
nchar(n)	char(n)
nvarchar(n)	varchar(n)
text	clob
unitext	nclob
binary(n)	binary(n)
varbinary(n)	varbinary(n)
image	blob
bit	tinyint
bigdatetime	timestamp

SAP ASE Datatype	HANA Datatype
bigtime	Not supported

These datatypes are allowed when you map remote HANA columns to local proxy table columns:

Remote HANA Datatype	SAP ASE Datatype
tinyint	tinyint
smallint	smallint
int	integer
integer	integer
bigint	bigint
decimal(precision, scale)	decimal(precision, scale)
real	real
double	double precision
float(precision)	double precision
date	date
time	time
seconddate	datetime
timestamp	bigdatetime
varchar(n)	varchar(n)
nvarchar(n)	univarchar(n)
alphanum(n)	univarchar(n)
char(n)	char(n)
nchar(n)	unichar(n)
varbinary(n)	varbinary(n)
clob	text
binary (n)	binary (n)
blob	image
nclob	unitext

Restrictions

CIS support for HANA includes some restrictions.

- CIS for HANA is supported only in SAP ASE servers that are configured for threaded kernel mode. You cannot use CIS for HANA in SAP ASE servers that are configured for process mode.
- CIS for HANA does not support **update** or **delete** commands that include a **where current of** *cursor_name* clause.
- CIS for HANA does not support sensitive cursors.
- CIS for HANA does not support large object (LOB) parameters.
- You cannot issue queries that include text pointers to the remote HANA server because HANA LOBs do not support text pointers.
- If you issue **create index...desc** to create an index on a proxy table, the **desc** parameter is not generated for the **create index** statement. Instead, the index in the remote HANA table is created with an **asc** parameter.
- **alter table** statements to remote tables include restrictions. You cannot:
  - Include partition clauses
  - Include referential integrity constraints
  - Add constraints
  - Replace columns
  - Change locking schemes
  - Add columns with **check** conditions
  - Add columns with unique or primary key constraints
- A single **alter table** command cannot include more than one of these parameters:
  - **add** *column_name*
  - **modify** *column_name*
  - **drop** *column_name*
- You must define a unique index on the base table for positioned **update** and **delete** statements, and the unique index must have at least one nonnullable column. SAP ASE issues an error message if this unique index is unavailable.
- You cannot use distributed transaction management (DTM) with two-phase commit on HANA proxy tables.
- You cannot include **save tran** and **rollback to savepoint** statements in transactions that involve HANA proxy tables.
- **rollback tran** operations issued on transactions involving HANA proxy tables roll back all statements up to the first statement in the current transaction. All statements after the **rollback tran** operation execute normally.
- You cannot issue **rollback trigger** on triggers that include HANA proxy tables.

- Search clauses can contain only the input parameters (and none of the resultant columns) for queries using proxy tables that are mapped to an external HANA procedure. For example, if you create this table:

  ```
  create existing table rpc_table
  (
     col1 int,
     col2 int
  ) external procedure at "odbchana..SYSTEM.MYPROC"
  ```

  Then issue the following query; the search results are not narrowed, because the query specifies `col1`:

  ```
  select * from rpc_table where col1 > 100
  ```
- You cannot use null as a value for input parameters for proxy tables that are mapped to an external HANA procedure. For example, this is not allowed:

  ```
  select * from rpc_table where _p1 = null
  ```
- CIS for HANA does not support bulk operations.
- Implicit conversions may cause a loss of precision for millisecond values for SAP ASE `bigdatetime` and `bigtime` datatypes that are mapped to a HANA `timestamp` datatype. SAP ASE stores the millisecond portion of `bigdatetime` and `bigtime` values using 6 digits (for example *hh:mi:ss.zzzzzz*). However, HANA stores the millisecond portion of `bigdatetime` and `bigtime` values in 7 digits in its `timestamp` datatype. Transmitting values using ODBC is precise up to 3 digits. Implicitly converting values from ODBC to the SAP ASE internal format may further depreciate the precision, because the available conversion formats support only up to 3 digits.
- You must fully pad binary values when you specify them in a search clause. In the first select statement below, the value of `c1` (0x123) is not fully padded, and therefore returns no results. The second select statement uses a fully padded value (0x012300) for `c1`, and thereby returns a valid result:

  ```
  1> create table cb(c1 binary(3)) at 'odbchana..SYSTEM.cb'
  2> go
  1> insert into cb values(0x123)
  2> go
  (1 row affected)

  1> select * from cb where c1 = 0x123
  2> go
  c 1
  --------
  (0 rows affected)

  3> select * from cb where c1 = 0x012300
  c1
  --------
  0x012300
  (1 row affected)
  ```

Restrictions for quoted identifiers when using CIS for HANA include:

- Object names in object definitions included with DDL statements must include at least one uppercase letter.
- Quoted identifiers for DML statements must include at least one uppercase letter.
- CIS for HANA raises an error for DDL statements that include an object definition that contains only lowercase characters. For example, this raises an error: `<hanaserver>..<schema>.table1`.
- SAP ASE creates object names in proxy tables without quotes. Consequently, SAP ASE cannot determine whether to add quotes to a quoted identifier when executing a DML statement if the remote quoted identifier contains only lowercase characters (for example `column1`). That is, SAP ASE creates the object name in the same way for `column1` as for `COLUMN1`, because SAP ASE defaults to lowercase. As a workaround, use trace flag 11242, which quotes all lowercase identifiers are quoted.
- CIS for HANA does not support bracketed identifiers (that is, [and]).

Relaxed Query Limits

SAP ASE version 16.0 increases some column and table restrictions.

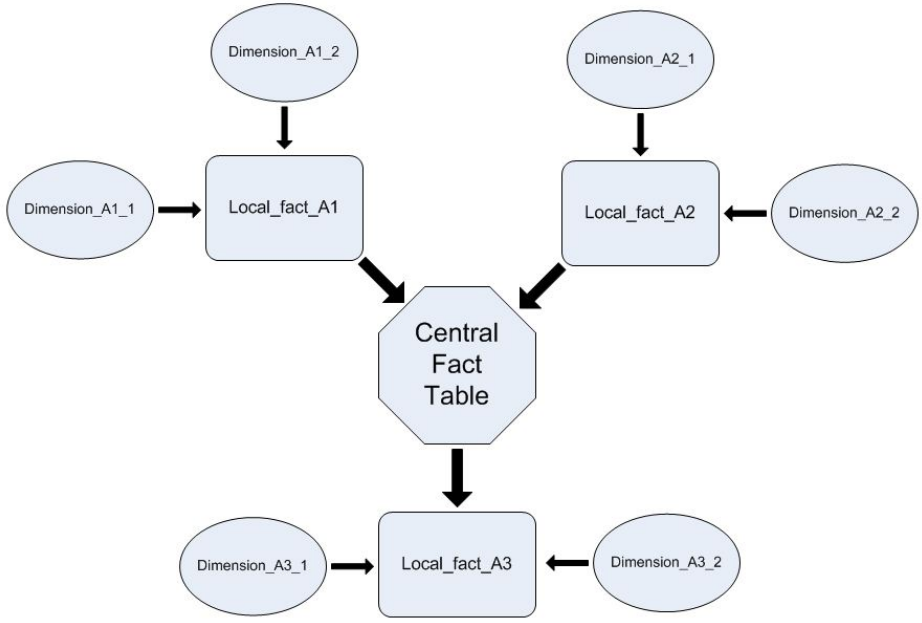
Limit	Limit in Versions Earlier Than 16.0	Limit in Versions 16.0 and Later
Number of tables in **select** queries	50 tables, 46 worktables	250 tables, 46 worktables
Number of subqueries in a **select** query	50	250
Maximum number of columns allowed in an **order by** clause	31	400

Query Plan Optimization with Star Joins

SAP ASE version 16.0 improves performance for star joins.

A star join is a commonly used data warehouse query that runs against a star schema database, which consists of a large table (also known as a fact table) surrounded by dimension tables. Fact tables typically include two types of columns: one that includes measurements, metrics, or facts about the business process, and another column that includes foreign keys to dimension tables. Dimension tables usually include descriptive attributes. For example, a fact table may include information about the number of hiking books sold by a particular author on a particular day. The corresponding dimension table includes the address, age, and gender of the author.

Star joins join the fact table with one or more dimension tables along the foreign keys. The star join may include filter predicates on the dimension tables (for example, **where** gender = 'M'), but it contains no joins between the dimension tables.



Snowflake schemas extend star joins by allowing each of the dimension tables to act as local fact tables and join with another set of dimension tables.

This star join joins the `Orders` fact table with the `Time`, `Customer`, and `Location` dimension tables:

```
select C.dattr, sum(O.measure1), count(*)
from Orders O
    join Time T on O.key_dim1 = T.key_col
    join Customer C on O.key_dim2 = C.key_col
    join Location L on O.key_dim3 = L.key_col
where T.dattr between 1 and 1000
  and C.dattr between 1001 and 2000
  and L.dattr between 901 and 1900
group by C.dattr
```

To improve performance for star join query evaluation these changes have been made in version 16.0:

- For parallel plans, the bloom filter probe moves below the `Xchg` operator.
- A new star join hint uses abstract plan to help the query processor better detect star join queries.
- The final plan of any star join is now forced to use bloom filter under the **use fact_table** hint.
- Hash joins and a join orders are forced based on the selectivities between fact and dimension tables for the star join query under the **use fact_table** hint. The query processor adjusts the query plan to provide the greatest benefit from the bloom filter.

The query optimizer uses star joins when appropriate. You need not configure SAP ASE to use star joins. However, SAP does recommend that you enable the **set join_bloom_filter** option and parallel query processing when using star joins.

Note: When you set the optimization goal to **allrows_dss**, the query processor enables **set join_bloom_filter** and parallel query processing.

Star Join Hint

SAP ASE version 16.0 introduces the **use fact_table** abstract plan hint, which specifies the central fact table in a star join query, and triggers special query plan optimization strategies for the query.

The syntax is:

```
PLAN '(use fact_table fact_table_name_or_alias_name)'
```

where:

- **fact_table** – indicates that the query includes a fact table from a star join.
- *fact_table_name_or_alias_name* – specifies the name or alias name used for the central fact table. In this example, `F` is the alias name:

  ```
  use fact_table 'F'
  ```

For example, this joins the `Orders` fact table to the `Time`, `Customer`, and `Location` dimension tables:

```
select C.dattr, sum(O.measure1), count(*)
from Orders O
```

```
join Time T on O.key_dim1 = T.key_col
join Customer C on O.key_dim2 = C.key_col
join Location L on O.key_dim3 = L.key_col
where T.dattr between 1 and 1000
and C.dattr between 1001 and 2000
and L.dattr between 901 and 1900
group by C.dattr
plan '(use fact_table O)'
```

The **use fact_table** hint makes these changes to the query processor's star query detection algorithm and plan generation:

- The detection algorithm starts with the *fact_table_alias_name*. This table is always the center of a snowflake schema.
- Allows **or** clauses.
- Allows joins between the fact table and the dimension tables to be multicolumn equijoins. You need not include an explicit constraint definition between primary and foriegn keys.
- If the query processor cannot locate a good plan for the nested loop join while it generates the query plan for the star join (typically due to missing indexes or less restrictive selectivities), it choses a hash join plan with a forced join order. The join order is based on the selectivities between the dimension and the fact tables. The higher the selectivity (that is, a lower ratio value), the closer the dimension tables are in that join order to the fact table.

The query processor ignores the **use fact_table** hint unless the query meets these conditions:

- The query has at least three tables.
- The query has no outer joins or nested subqueries.
- The hinted fact table is the largest joining table within its query block.
- The dimension tables and the fact table are joined with equijoins.
- There are no join predicates between the dimension tables.

Star Join Query Plans Under the use fact_table Hint

The **use fact_table** abstract plan hint allows the query processor to chose a parallel hash join plan for the star join query. Parallel plans enable the query processor to push bloom filter probings (which allow for faster joins between dimension and fact tables) below the `EXCHANGE` operator, further reducing the number of qualifying rows from fact tables.

Versions of SAP ASE earlier than 16.0 included the bloom filter above the `EXCHANGE` operator in **sp_showplan** output.

This example includes a parallel plan under the hint, which includes table `F` as the hinted central fact table:

```
QUERY PLAN FOR STATEMENT 1 (at line 1).
Optimized using Parallel ModeExecuted in parallel by coordinating
process and 66 worker processes.
```

```
STEP 1
The type of query is SELECT.

29 operator(s) under root

|ROOT:EMIT Operator (VA = 29)
|
| |HASH VECTOR AGGREGATE Operator (VA = 28)
| | GROUP BY
| | Evaluate Grouped COUNT AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable10 for internal storage.
| | Key Count: 3
| |
| | |HASH JOIN PROBE Operator (VA = 27) (Join Type: Inner Join)
| | | Using Worktable1 for internal storage.
| | | Key Count: 1
| | |
| | | |HASH JOIN BUILD Operator (VA = 1) (Join Type: Inner Join)
| | | | Using Worktable1 for internal storage.
| | | | Key Count: 1
| | | |
| | | | |SCAN Operator (VA = 0)
| | | | | FROM TABLE
| | | | | /B49/DBENCH01P
| | | | | DP
| | | | | Index : /B49/DBENCH01P~010
| | | | | Forward Scan.
| | | | | Positioning by key.
| | | | | Keys are:
| | | | | SID_0CHNGID ASC
| | | | | SID_0RECORDTP ASC
| | | | | SID_0REQUID ASC
| | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index leaf pages.
| | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | With LRU Buffer Replacement Strategy for data pages.
| | |
| | | |HASH JOIN PROBE Operator (VA = 26) (Join Type: Inner Join)
| | | | Using Worktable2 for internal storage.
| | | | Key Count: 1
```

```
| | | |
| | | | |HASH JOIN BUILD Operator (VA = 3) (Join Type: Inner Join)
| | | | | Using Worktable2 for internal storage.
| | | | | Key Count: 1
| | | | |
| | | | | |SCAN Operator (VA = 2)
| | | | | | FROM TABLE
| | | | | | /B49/DBENCH01U
| | | | | | DU
| | | | | | Index : "/B49/DBENCH01U~020"
| | | | | | Forward Scan.
| | | | | | Positioning at index start.
| | | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index leaf pages.
| | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | |
| | | | |HASH JOIN PROBE Operator (VA = 25) (Join Type: Inner Join)
| | | | | Using Worktable3 for internal storage.
| | | | | Key Count: 2
| | | | |
| | | | | |HASH JOIN BUILD Operator (VA = 5) (Join Type: Inner Join)
| | | | | | Using Worktable3 for internal storage.
| | | | | | Key Count: 2
| | | | | |
| | | | | | |SCAN Operator (VA = 4)
| | | | | | | FROM TABLE
| | | | | | | /B49/DBENCH01T
| | | | | | | DT
| | | | | | | Index : /B49/DBENCH01T~20
| | | | | | | Forward Scan.
| | | | | | | Positioning by key.
| | | | | | | Keys are:
| | | | | | | SID_0CALMONTH ASC
| | | | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | | | With LRU Buffer Replacement Strategy for index leaf
pages.
| | | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | |
| | | | | |HASH JOIN PROBE Operator (VA = 24) (Join Type: Inner Join)
| | | | | | Using Worktable4 for internal storage.
| | | | | | Key Count: 1
| | | | | |
| | | | | | |HASH JOIN BUILD Operator (VA = 7) (Join Type: Inner Join)
| | | | | | | Using Worktable4 for internal storage.
| | | | | | | Building pushdown bloom filter (bv7)
| | | | | | | Key Count: 1
| | | | | | |
| | | | | | | |SCAN Operator (VA = 6)
| | | | | | | | FROM TABLE
| | | | | | | | /B49/XCUSTOMER
| | | | | | | | X1
| | | | | | | | Table Scan.
| | | | | | | | Forward Scan.
| | | | | | | | Positioning at start of table.
```

```
| | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | | |
| | | | | | |HASH JOIN PROBE Operator (VA = 23) (Join Type: Inner
Join)
| | | | | | | Using Worktable5 for internal storage.
| | | | | | | Key Count: 1
| | | | | | | |
| | | | | | | |HASH JOIN BUILD Operator (VA = 9) (Join Type: Inner
Join)
| | | | | | | | Using Worktable5 for internal storage.
| | | | | | | | Building pushdown bloom filter (bv9)
| | | | | | | | Key Count: 1
| | | | | | | | |
| | | | | | | | |SCAN Operator (VA = 8)
| | | | | | | | | FROM TABLE
| | | | | | | | | /B49/DBENCH011
| | | | | | | | | D1
| | | | | | | | | Table Scan.
| | | | | | | | | Forward Scan.
| | | | | | | | | Positioning at start of table.
| | | | | | | | | Using pushdown bloom filter (bv7).
| | | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | | |
| | | | | | |EXCHANGE Operator (VA = 22) (Merged)
| | | | | | |Executed in parallel by 64 Producer and 1 Consumer
processes.

| | | | | | | |
| | | | | | | |EXCHANGE:EMIT Operator (VA = 21)
| | | | | | | | |
| | | | | | | | |HASH JOIN PROBE Operator (VA = 20) (Join Type:
Inner Join)
| | | | | | | | | Using Worktable9 for internal storage.
| | | | | | | | | Key Count: 1
| | | | | | | | | |
| | | | | | | | | |EXCHANGE Operator (VA = 13) (Replicated)
| | | | | | | | | |Executed in parallel by 1 Producer and 64
Consumer processes.

| | | | | | | | | | |
| | | | | | | | | | |EXCHANGE:EMIT Operator (VA = 12)
| | | | | | | | | | | |
| | | | | | | | | | |HASH JOIN BUILD Operator (VA = 11) (Join Type:
Inner Join)
| | | | | | | | | | | Using Worktable6 for internal storage.
| | | | | | | | | | | Building pushdown bloom filter (bv11)
| | | | | | | | | | | Key Count: 1
| | | | | | | | | | |
| | | | | | | | | | | |SCAN Operator (VA = 10)
| | | | | | | | | | | | FROM TABLE
| | | | | | | | | | | | /B49/SSALESORG
| | | | | | | | | | | | S1
| | | | | | | | | | | | Index : "/B49/SSALESORG~0"
| | | | | | | | | | | | Forward Scan.
```

```
| | | | | | | | | | | | | | Positioning by key.
| | | | | | | | | | | | | | Keys are:
| | | | | | | | | | | | | | /B49/S_SALESORG ASC
| | | | | | | | | | | | | | Using I/O Size 16 Kbytes for index leaf
pages.
| | | | | | | | | | | | | | With LRU Buffer Replacement Strategy for
index leaf pages.
| | | | | | | | | | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | | | | | | | | | With LRU Buffer Replacement Strategy for
data pages.
| | | | | | | | | | |
| | | | | | | | | | |HASH JOIN PROBE Operator (VA = 19) (Join Type:
Inner Join)
| | | | | | | | | | Using Worktable8 for internal storage.
| | | | | | | | | | Key Count: 1
| | | | | | | | | | |
| | | | | | | | | | |EXCHANGE Operator (VA = 17) (Replicated)
| | | | | | | | | | |Executed in parallel by 1 Producer and 64
Consumer processes.

| | | | | | | | | | | |
| | | | | | | | | | | |EXCHANGE:EMIT Operator (VA = 16)
| | | | | | | | | | | |
| | | | | | | | | | | |HASH JOIN BUILD Operator (VA = 15) (Join
Type: Inner Join)
| | | | | | | | | | | | Using Worktable7 for internal storage.
| | | | | | | | | | | | Building pushdown bloom filter (bv15)
| | | | | | | | | | | | Key Count: 1
| | | | | | | | | | | |
| | | | | | | | | | | |SCAN Operator (VA = 14)
| | | | | | | | | | | | FROM TABLE
| | | | | | | | | | | | /B49/DBENCH013
| | | | | | | | | | | | D3
| | | | | | | | | | | | Table Scan.
| | | | | | | | | | | | Forward Scan.
| | | | | | | | | | | | Positioning at start of table.
| | | | | | | | | | | | Using pushdown bloom filter (bv11).
| | | | | | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | | | | | With LRU Buffer Replacement Strategy for
data pages.
| | | | | | | | | | |
| | | | | | | | | | |SCAN Operator (VA = 18)
| | | | | | | | | | | FROM TABLE
| | | | | | | | | | | /B49/FBENCH01
| | | | | | | | | | | F
| | | | | | | | | | | Index : /B49/FBENCH01~060
| | | | | | | | | | | Forward Scan.
| | | | | | | | | | | Positioning at index start.
| | | | | | | | | | | Using pushdown bloom filter (bv15, bv9).
| | | | | | | | | | Executed in parallel with a 64-way partition
scan.
| | | | | | | | | | Using I/O Size 128 Kbytes for index leaf pages.
| | | | | | | | | | With LRU Buffer Replacement Strategy for index
leaf pages.
| | | | | | | | | | Using I/O Size 128 Kbytes for data pages.
```

```
| | | | | | | | | | | With MRU Buffer Replacement Strategy for data
pages.
```

This example includes a serial plan under the hint:

```
QUERY PLAN FOR STATEMENT 1 (at line 1).
Optimized using Serial Mode


STEP 1
The type of query is SELECT.

8 operator(s) under root

|ROOT:EMIT Operator (VA = 8)
|
| |HASH VECTOR AGGREGATE Operator (VA = 7)
| | GROUP BY
| | Evaluate Grouped COUNT AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable4 for internal storage.
| | Key Count: 1
| | |
| | |HASH JOIN Operator (VA = 6) (Join Type: Inner Join)
| | | Using Worktable3 for internal storage.
| | | Building pushdown bloom filter (bv6)
| | | Key Count: 1
| | | |
| | | |SCAN Operator (VA = 0)
| | | | FROM TABLE
| | | | Location
| | | | L
| | | | Table Scan.
| | | | Forward Scan.
| | | | Positioning at start of table.
| | | | Using I/O Size 2 Kbytes for data pages.
| | | | With LRU Buffer Replacement Strategy for data pages.
| | | |
| | | |HASH JOIN Operator (VA = 5) (Join Type: Inner Join)
| | | | Using Worktable2 for internal storage.
| | | | Building pushdown bloom filter (bv5)
| | | | Key Count: 1
| | | | |
| | | | |SCAN Operator (VA = 1)
| | | | | FROM TABLE
| | | | | Customer
| | | | | C
| | | | | Table Scan.
| | | | | Forward Scan.
| | | | | Positioning at start of table.
| | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | |
| | | | |HASH JOIN Operator (VA = 4) (Join Type: Inner Join)
| | | | | Using Worktable1 for internal storage.
| | | | | Building pushdown bloom filter (bv4)
```

```
| | | | | Key Count: 1
| | | | |
| | | | | |SCAN Operator (VA = 2)
| | | | | | FROM TABLE
| | | | | | Time
| | | | | | T
| | | | | | Table Scan.
| | | | | | Forward Scan.
| | | | | | Positioning at start of table.
| | | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | |
| | | | | |SCAN Operator (VA = 3)
| | | | | | FROM TABLE
| | | | | | Orders
| | | | | | O
| | | | | | Table Scan.
| | | | | | Forward Scan.
| | | | | | Positioning at start of table.
| | | | | | Using pushdown bloom filter (bv4, bv5, bv6).
| | | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | | With LRU Buffer Replacement Strategy for data pages.
```

Query Performance Improvements

SAP ASE version 16.0 introduces improvements to queries.

Dynamic Thread Assignment

SAP ASE version 16.0 and later uses dynamic instead of static thread assignments.

Dynamic thread assignment allows SAP ASE to execute parallel query plans faster and with fewer resources. SAP ASE applies dynamic thread assignment to parallel lava query plans that are generated for **select** queries. However, these commands continue to use static thread assignment:

- **select into**
- **reorg** commands that include a data copy
- **alter table** commands that include a data copy
- **create index**

Dynamic thread assignment improves performance by:

- Executing query plans in parallel with fewer threads (static thread assignment requires serial query execution).
- Executing dynamic load balancing between worker threads: When there are fewer threads executing the query plan than there are work units, the threads that execute the smallest work units complete their task more quickly, freeing themselves to execute the remaining available work units while the threads executing larger work units complete their tasks.
- Using existing semantic partitioning in joins more effectively. Dynamic thread assignment allows a partition-to-partition join when joining two tables that are already partitioned on the joining column, as opposed to static thread assignment, which joins all the partitions of the outer table to all of the partitions of the inner table in one operation. Dynamic thread assignment allows a single worker thread to join the first partition of the outer table to the first partition of the inner table, then reexecutes the query plan fragment to join the second partition, and so on.

Dynamic thread assignment is enabled when you configure SAP ASE for parallelism with the **number of worker processes** and **max parallel degree** parameters. See *Setting Configuration Parameters* in the *System Administration Guide, Volume 1*.

SORT Operator Performance Improvement

SAP ASE version 16.0 and later improves the performance on certain parallel queries that include `SORT` operators.

To benefit from the changes, the parallel query must include:

- A `SORT` operator that appears on top of the `Exchange` operator
- `Exchange` operator must be a `Replicated Exchange` with a single producer and multiple consumers.

If the query does not meet this criteria, the query processor uses the standard `SORT` operator included with SAP ASE versions earlier than 16.0 (also called the DEFAULT sort).

This example is from a version of SAP ASE earlier than 16.0, and does not include the `SORT` operator changes:

```
|   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join) (VA = 5)
|   |   |   |   | Using Worktable2 for internal storage.
|   |   |   |   |  Key Count: 1
|   |   |   |   |  Key Ordering: ASC
|   |   |   |   |
|   |   |   |   |   |SORT  Operator (VA = 3)
|   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |
|   |   |   |   |   |   |EXCHANGE Operator (VA = 2) (Replicated)
|   |   |   |   |   |   |Executed in parallel by 1 Producer and 8
Consumer processes.
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 1)
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |SCAN Operator (VA = 0)
|   |   |   |   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |   |   |   |  t2
|   |   |   |   |   |   |   |   |  Table Scan.
|   |   |   |   |   |   |   |   |  Forward Scan.
|   |   |   |   |   |   |   |   |  Positioning at start of table.
|   |   |   |   |   |   |   |   |  Using I/O Size 16 Kbytes for data
pages.
|   |   |   |   |   |   |   |   |  With LRU Buffer Replacement Strategy
for data pages.
```

However, SAP ASE version 16.0 and later moves the build part of the `SORT` operator below the `Exchange Operator` and keeps the sort-table reading part above the `Exchange`, which results in this query execution plan for the same query

```
|   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join) (VA = 6)
|   |   |   |   | Using Worktable3 for internal storage.
|   |   |   |   |  Key Count: 1
|   |   |   |   |  Key Ordering: ASC
|   |   |   |   |
|   |   |   |   |   |SORT (GETSORTED) Operator (VA = 4)
```

```
|   |   |   |   |   | Using Worktable2 for internal storage.
|   |   |   |   |   |
|   |   |   |   |   |   |EXCHANGE Operator (VA = 3) (Replicated)
|   |   |   |   |   |   |Executed in parallel by 1 Producer and 8
Consumer processes.
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 2)
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |SORT (SORTBUILD) Operator (VA = 1)
|   |   |   |   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |SCAN Operator (VA = 0)
|   |   |   |   |   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |   |   |   |   |  t2
|   |   |   |   |   |   |   |   |   |  Table Scan.
|   |   |   |   |   |   |   |   |   |  Forward Scan.
|   |   |   |   |   |   |   |   |   |  Positioning at start of table.
|   |   |   |   |   |   |   |   |   |  Using I/O Size 16 Kbytes for data
pages.
|   |   |   |   |   |   |   |   |   |  With LRU Buffer Replacement
Strategy for data pages.
```

The benefits of the new parallel `SORT` operator are that it:

- Splits the `SORT` operator into two operators: `SORTBUILD` and `GETSORTED`. A single sort table built by the `SORTBUILD` is used by multiple instances of `GETSORTED` for a parallel reading of sorted rows, reducing the I/Os. The DEFAULT `SORT` operator plan creates multiple copies of `SortTables` (8 `Consumer processes` in the first example).
- Sends a single row containing meta data from the `SORTBUILD` operator to multiple instances of `GETSORTED` operator, providing a reference for accessing the sort-table directly, which avoids copying and propagating many rows via `Exchange`.

The parallel `SORT` operator includes these restrictions:

- Does not support `Repartitioned Exchange`. However, `SORT` operators that occur above `Repartitioned Exchange` use the DEFAULT sort in the query execution plan.
- Although abstract plans do not explicitly support forcing `GETSORTED` and `SORTBUILD`, you can force these plans by making sure the `SORT` operator appears above the `Replicated Exchange`.
- SAP ASE implicitly disables dynamic thread assignment (DTA) for the query execution plan that contains `GETSORTED` and `SORTBUILD` operators.

Hash Join Operator Performance Improvement

SAP ASE versions 16.0 and later implement performance improvements and a reduction in resource usage for certain parallel query plans involving `HASH JOIN` operators.

A `HASH JOIN` operation includes two steps:

1. Building a hash table containing the joining columns of the outer stream of rows to be joined, and
2. Probing the hash table with the joining column values of each of the rows in the inner stream to be joined

SAP ASE 16.0 replaces the `HASH JOIN` operator with the `HASH PROBE` and the `HASH BUILD` operators, and includes the replicated `EXCHANGE` operator between these operators. The `HASH BUILD` operator builds the hash table and the `HASH PROBE` operator reads the inner stream and probes the hash table to find matching rows (in earlier releases of SAP ASE, the `HASH JOIN` operator performed both these steps).

A single worker thread executes the `HASH BUILD` operator, building a single hash table. The query engine passes this hash table through memory pipes to all producers executing the `HASH PROBE` operator. These producers share this hash table and probe it for matches to the joining columns in their inner streams. Earlier releases of SAP ASE required multiple producers to execute the `HASH JOIN` operator.

This is an example of **showplan** output for a query plan using the `HASH JOIN` operator:

```
|
|   |EXCHANGE Operator (VA = 6) (Merged)
|   |Executed in parallel by 4 Producer and 1 Consumer processes.

|   |
|   |   |EXCHANGE:EMIT Operator (VA = 5)
|   |   |
|   |   |   |HASH JOIN Operator (VA = 4) (Join Type: Inner Join)
|   |   |   | Using Worktable1 for internal storage.
|   |   |   |  Key Count: 1
|   |   |   |
|   |   |   |   |EXCHANGE Operator (VA = 2) (Replicated)
|   |   |   |   |Executed in parallel by 1 Producer and 4 Consumer
processes.

|   |   |   |   |
|   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 1)
|   |   |   |   |   |
```

A query plan must include these attributes for the query processor to use the `HASH JOIN` improvements:

- The query processor executes the `HASH JOIN` operator (VA = 4) in parallel (the example above includes four `Producers` in the `EXCHANGE Operator (VA = 6).`
- The left child `Operator` of the `HASH JOIN Operator` is a `Replicated EXCHANGE Operator (VA = 2)` with a single `Producer`

If the query does not meet this criteria, the query processor uses the single HASH JOIN operator.

The benefits of the SAP ASE 16.0 query plan are that it:

- Avoids the performance degradation caused by copying all of the rows through the memory pipes to the producers executing the `HASH JOIN` operator. Instead, the query engine passes a single row containing a reference to the hash table built by the `HASH PROBE` operator through the pipes.
- Does not require the extra memory and disk resources needed by each producer to build its own hash table. Instead, it shares a single hash table.

This is an example of showplan output for query plans using the improved hash join processing:

```
|
|   |EXCHANGE Operator (VA = 7) (Merged)
|   |Executed in parallel by 4 Producer and 1 Consumer processes.

|   |
|   |   |EXCHANGE:EMIT Operator (VA = 6)
|   |   |
|   |   |   |HASH JOIN PROBE Operator (VA = 5) (Join Type: Inner Join)
|   |   |   | Using Worktable2 for internal storage.
|   |   |   |  Key Count: 1
|   |   |   |
|   |   |   |   |EXCHANGE Operator (VA = 3) (Replicated)
|   |   |   |   |Executed in parallel by 1 Producer and 4 Consumer
processes.

|   |   |   |   |
|   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 2)
|   |   |   |   |   |
|   |   |   |   |   |   |HASH JOIN BUILD Operator (VA = 1) (Join Type:
Inner Join)
|   |   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |   |  Key Count: 1
|   |   |   |   |   |   |
```

Full-Text Auditing

SAP ASE versions 16.0 and later support full-text DML auditing, printing parameter names and values with sensitive parameters masked when you enable the DML auditing options (including *table_access* and *view_access*).

For information about full text auditing see, *Security Administration Guide > Auditing > Manage the Audit System > Configure the Audit System > Set Global Auditing Options > Auditing DML Statements*.

Auditing for Authorization Checks Inside Stored Procedures

The audit option `sproc_auth` enables auditing for authorization checks that are performed inside system stored procedures.

Granular Permissions	Event
Enabled	146
Disabled	80

The audit event 80 is audited when the audit option `security` is enabled, or when the audit option `sproc_auth` is enabled. The audit event 146 is only audited when the option `sproc_auth` is enabled.

Replacing Object Definitions

Replace existing compiled object definitions with new definitions while preserving the original names, object IDs, security attributes, such as auditing options and permissions, and replication attributes.

The **create or replace** functionality creates a new object if it does not exist, or replaces an existing object with the same name. The **or replace** clause implicitly drops and re-creates an existing object of the same name and type within the database, changing the definition of the object, while preserving the existing security and replication attributes. If the text of the compiled object was hidden before it was replaced, it will remain hidden after being replaced. No new keywords have been introduced by this feature.

The **or replace** clause has been added to these commands:

- **create default**
- **create function**
- **create function (SQLJ)**
- **create procedure**
- **create procedure (SQLJ)**
- **create rule**
- **create trigger**
- **create view**

The or replace functionality is supported only for objects that do not contain data.

If the object is in use while being replaced, error 3702 is raised:

```
"Cannot drop or replace the %S_MSG '%.*s' because it is currently in
use."
```

When an object is replaced, SAP ASE replaces its definition in the following system tables: `sysprocedures`, `syscomments`, `sysdepends`, and `syscolumns`. Some fields in the `sysobjects` table are also updated. The query tree for the object is normalized before being replaced in `sysprocedures`.

The replaced object may be used in other object definitions. SAP ASE recompiles the replaced object when it is used, however, in some cases, you may need to replace the calling object when the interface of the replaced object does not match with that used in the calling object. You can run **sp_depends** on the replaced object to verify whether there are calling objects and then replace them. For details, see the section *Objects Dependent on Replaced Objects* in *create procedure*, *create view*, and *create function*.

With granular permissions enabled or disabled, you must be the object owner to replace a compiled object. You cannot replace a compiled object by impersonating the object owner

through an alias or **setuser**. However, if you are the owner through **set proxy**, you can replace a compiled object.

Note: The **create or replace** functionality performs an implicit **drop** followed by **create** in the same transaction. Because of this, additional transaction log space is required. If you use **create or replace** instead of dropping an object and then creating the object, you may need to increase the size of the transaction log.

See also

- *create default* on page 121
- *create function* on page 125
- *create function (SQLJ)* on page 128
- *create procedure* on page 132
- *create procedure (SQLJ)* on page 135
- *create rule* on page 137
- *create trigger for or replace* on page 145
- *create view* on page 148

Install Script Changes

Stored procedures used in install scripts have been changed to use **create or replace**.

In releases earlier than SAP ASE 16.0, stored procedures used in install scripts were dropped and re-created. As of SAP ASE 16.0, stored procedures used in these install scripts use the **or replace** functionality:

installmaster	installmodel	installmsgsvss	installpcidb
installjconnect	installcommit	installdbccalt	installdbccdb
installupgrade	installjsb	installsecurity	installoledb
installodbc	installhasvss	installpubs2	installpubs3

Data and Log Segment Changes

Changes to the size of data and log segments for various databases have been implemented.

For **create or replace**, devices need to be large enough to account for extra space requirements.

Default database size changes are as follows:

Database	Page size	Default size earlier than 16.0	Default size 16. 0 and later
master	2KB	13MB	18MB
	4KB	26MB	26MB
	8KB	52MB	52MB
	16KB	104MB	104MB
sybpcidb	2KB	24MB	48MB
	4KB	48MB	96MB
	8KB	96MB	192MB
	16KB	192MB	384MB

Database	Default size earlier than 16.0	Default size 16. 0 and later
sybsystemprocs	172MB	196MB
sybmgmtdb	75MB	76MB
dbccdb	5MB	33MB
dbcalt	5MB	33MB

For more information about default values and minimum requirements for configuring an SAP ASE server, see *Installation Guide > Installing SAP ASE > Minimally Configuring an SAP ASE Server*.

Query Plan and Execution Statistics in HTML

A new dynamic thread assignment (DTA) model enhances the use of worker threads.

Version 15.7 SP100 introduced a feature to generate a graphical query plan in HTML format for viewing data in a Web browser using the static thread assignment (STA) worker thread model. Using STA, there is a direct relationship between plan fragments and worker threads: as many plan fragments are generated as there are worker threads assigned to execute the query. The assignment of plan fragments to worker threads is static and unique: a worker thread can only execute the plan fragment assigned. Using this model, the HTML representation prints each thread with the details about its single plan fragment execution.

In version 16.0, the DTA worker thread model has been implemented. With the introduction of DTA, the Query Plan and Execution Statistics in HTML feature has been updated to correctly reflect the new parallel execution model. See *Query Performance Improvements* for more information about the DTA worker thread model.

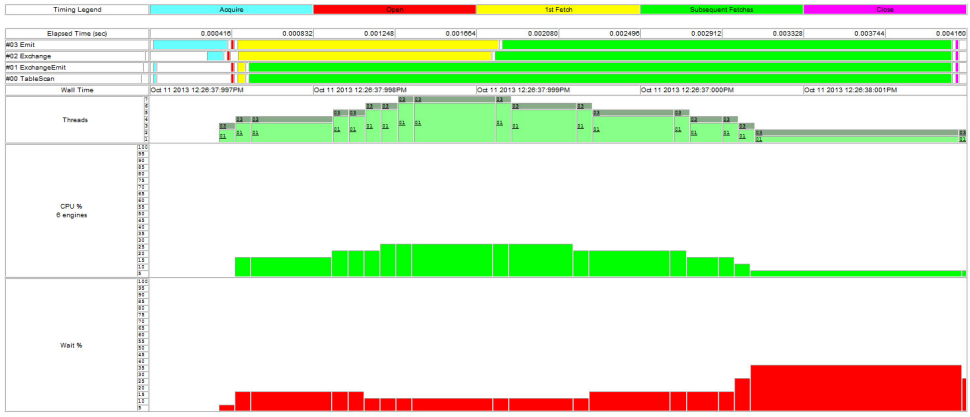
The HTML representation reports the execution statistics per plan fragment execution (or work units), allowing several executions of the same plan fragment to be reported separately. To better identify the work unit execution following the new DTA model, for each plan fragment execution, the output is indicated by "Work unit execution" and provides the SPID for the thread and additional values such as the Root operator identifier (for the plan fragment) and producer ID.

This is an example of a query executed in parallel using several worker threads.

Adaptive Server Enterprise ® Query Plan
Query: [spid: 45]
Version: Adaptive Server Enterprise

Query Tree

Query Timings



Query Text

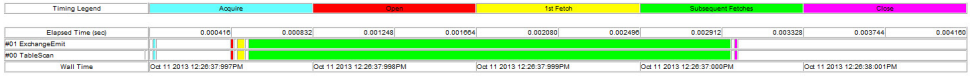
```
select * from RA2 (parallel 6)
```

Threads

SPID	Thread ID	Rows produced
34	2686997	168
35	2656924	168
36	2424851	168
37	2293778	166
38	2162705	162
39	2031632	168

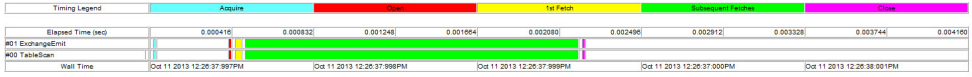
Work unit execution:
Thread SPID: 34 Root operator: 1 Producer ID: 0

Query Timings



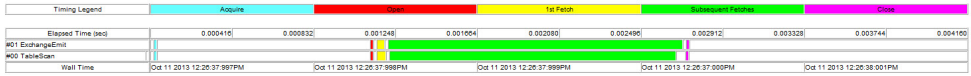
Work unit execution:
Thread SPID: 35 Root operator: 1 Producer ID: 1

Query Timings



Work unit execution:
Thread SPID: 36 Root operator: 1 Producer ID: 2

Query Timings



See also

- *Query Performance Improvements* on page 39

Option for Prefix of Generated File

The SET STATISTICS QUERY_NAME_HTML command helps differentiate or identify files that are related to multiple executions of the same query.

When given a query name, that query name is used as prefix for the internally generated file where the HTML output is written.

```
SET STATISTICS QUERY_NAME_HTML [queryname | ON | OFF]
```

Index Compression

Index compression in a relational database allows more-efficient data storage, reduced memory consumption, and improved performance due to lower I/O demands.

Index compression supports:

- Index leaf page compression
- Both DOL and APL index leaf page formats
- Compression at the table, index, and local index partition levels

You can enable or disable index compression at the server level, or at the session level, using **sp_configure** and **set compression**.

To specify index compression at the table, index, or local index partition levels, use these commands:

- **create table**
- **alter table**
- **create index**
- **alter index**
- **select into**

Specifying compression at the index level overrides index compression specified at the table level. Local index partition level specification overrides index-level specification.

Support

- **reorg rebuild** has been extended to support index compression.
- DML has been extended to support index compression.
- Index compression is supported by triggers.
- Data that has been bulk-copied into a table with indexes defined as compressed, is automatically compressed.
- Both **dbcc checktable** and **dbcc checkstorage** can check the integrity of indexes. These commands have been enhanced to check the integrity of each page by scanning all pages of a table. **dbcc checktable** has greater checking capacity because more information is checked.
- **dump database**, **dump transaction**, and **load transaction** are allowed if this database contains tables with a compressed index. However, XPDL is not allowed if there are any tables with compressed indexes.

Replication indexes are always created as uncompressed, even when compression is specified for all indexes during table creation.

APL-clustered indexes on index-compressed tables are not supported.

Unique indexes with only one column are not compressed.

Enabling Index Compression

Enable index compression at the server level or at the database level.

Enabling Index Compression at the Server Level

To enable compression on all indexes in all databases on a server, set index compression at the server level.

The syntax is:

```
sp_configure "enable compression", 0 | 1
```

The default value is 0.

If index compression is not enabled, an error is raised when you attempt to create an index compressed table or index.

Enabling Index Compression at the Session Level

To enable compression for all indexes in all databases of a session, set index compression at the database level.

The syntax is:

```
set {compression
   [= {default | ON | OFF} ]
   |index_compression
   [= {default | ON | OFF} ] }
```

The default value is off. This command affects only leaf rows that are built for compressed indexes after the command is executed.

- After **set index_compression** is set to off, all rows that are newly inserted into compressed index are not index compressed.
- After **set index_compression** is set to on, all rows that are newly inserted into compressed index are index compressed.

Creating an Index Compressed Table

To designate index compression for a table, use the **create table** or **select into** commands.

Any table or index, including temporary tables, may be designated for index compression, except of system catalogs and worktables.

Specifying compression at the index level overrides index compression specified at the table level. Local index partition level specification overrides the index-level specification.

*Using the **create table** Command*

The **with index_compression** clause provides these compression options:

- NONE – indexes on the specified table are not compressed. Indexes that are specifically created with **index_compression = PAGE** are compressed.
- PAGE – all indexes on the specified table are compressed. Indexes that are specifically created with **index_compression = NONE** are not compressed.

If compression has not been specified anywhere in the table DDL, indexes for the table are not compressed.

*Using the **select into** Command*

Use **select into** to create an index compressed table by selecting an existing table. The syntax for the **with index_compression** cause is the same as for the **create table** command.

See also

- *create table for Index Compression* on page 140
- *select into* on page 155

Creating a Compressed Index

To designate index compression for a index or local index partition level, use the **index_compression** clause.

Only leaf pages are compressed. Compressed and uncompressed index rows may coexist on a single index leaf page. If compression has not been specified anywhere in the table or index DDL, then the indexes are not compressed. APL-clustered indexes are not supported by index compression. Unique indexes that have only one column are not compressed.

Specifying compression at the index level overrides index compression specified at the table level. Local index partition level specification overrides the index-level specification.

*Using the **create index** Command*

The **WITH index_compression** clause provides these compression options:

- NONE – the index page for the specified index is not compressed. Local index partitions that are specifically created with **index_compression = PAGE** are compressed.
- PAGE – when the page is full, existing index rows are compressed using the page prefix compression. When a row is added, a check is performed to determine whether the row is suitable for compression.

See also

- *create index* on page 130

Changing the Compression State

To change the compression state of the table for future index inserts or updates, use **alter table** or **alter index**.

Existing index pages are not affected, whether or not they are compressed. To change the compression state of a table, you must have exclusive access to the table.

Changing the local index partition's compression state affects only index rows that are newly inserted or updated in the partition.

The default behavior for newly created indexes depends on the table's compression setting:

- For index-compressed tables, index compression is set to on for newly created indexes.
- For index-uncompressed tables, newly created indexes remain uncompressed.

*Using the **alter table** Command*

The **alter table** command permits many combinations of schema modifications and property modifications. Some of these commands require only a catalog update, whereas others need data movement, along with the rebuilding of any existing indexes. If indexes require rebuilding, and index compression is set to on, index pages are compressed as part of the index rebuilding. After index rebuilding, the resulting index contains compressed or uncompressed index rows according to the index compression state.

The **set index_compression** clause specifies the index compression to be enabled or disabled on the table, index, or local index partition. If table is modified to be index compressed, newly created indexes are compressed.

The **modify partition** *partition_name* clause names local index partitions for which the compression state is being modified as specified by the set compression clause that follows.

*Using the **alter index** Command*

The permissions for **alter index** defaults to the index owner and cannot be transferred except to the database owner, who can impersonate the index owner by running the **setuser** command. A system administrator can also alter user indexes.

Limitations

APL-clustered indexes are not supported by index compression.

Unique indexes that have only one column are not compressed.

Removing Compression from Indexes

To remove compression from indexes, the indexes must be dropped and then re-created with **set index_compression off**.

See also

- *alter table for Index Compression* on page 116
- *alter index* on page 116

SAP JVM Support

SAP ASE uses SAP JRE to support Java applications.

By default, the SAP JRE is installed in:

```
$SYBASE/shared/SAPJRE-7_*
```

The installer automatically sets the *SAP_JRE7*, *SAP_JRE7_32*, and *SAP_JRE7_64* environment variables.

Note: (IBM AIX only) You must set the data size resource limit to `umlimited` when using any Java application:

```
limit datasize unlimited
```

See your operating system documentation.

Full Database Encryption

SAP ASE version 16.0 introduces the ability to fully encrypt databases.

Earlier versions of SAP ASE allowed you to encrypt columns. With version 16.0, you can fully encrypt entire databases, providing protection for an entire database. When you fully encrypt a database, all of its data, indexes, and transaction logs become encrypted. This encryption is transparent, so that users can perform operations on tables, indexes, and so on, as usual, without noticing any differences.

SAP ASE encrypts data at the page level. Once your database is set up as encrypted, the encryption and decryption process is automatic. Data is encrypted just before the page is written into disk, and decrypted as soon as the data page is loaded into memory.

Full Database Encryption Versus Encrypted Columns

Encrypt entire databases, or only columns, depending on your needs.

SAP ASE authentication and access control mechanisms ensure that only properly identified and authorized users can access data. Data encryption further protects sensitive data against theft and security breaches.

While both encrypted columns and fully encrypted databases allow you to comply with security and privacy requirements, the different usages may make one feature easier to deploy than the other. Consider using:

- Encrypted columns when you can easily identify which columns contain sensitive data.
- Encrypted databases when you must perform range searches over sensitive data columns, and when you lack the knowledge of the data model and cannot identify sensitive data columns (for example, in packaged applications that include thousands of tables). In addition, the definition of sensitive data (such as personal information) differs among different locations (such as states or countries); encrypting an entire database can allow you to satisfy these various data security requirements.

Creating the Database Encryption Key

The database encryption key is a 256-bit symmetric key that is created in the master database and used to encrypt a database.

Prerequisites

Before you can create a database encryption key (DEK):

- Verify that you have a valid SAP ASE encryption feature license (ASE_ENCRYPTION)
- Set the **enable encrypted columns** configuration parameter
- Create a master key and optionally, a dual master key in the `master` database; these protect the database encryption key. See *Using Database-Level Master and Dual Master Keys* in the *Encrypted Columns Users Guide*.
- Ensure that you have the appropriate privileges:
  - If granular permission is enabled, a system permission called `manage database encryption key` is required to create the key.
  - If granular permission is disabled, you must have sso_role, keycustodian_role, or **create encryption key** permission.

Task

Use the **create encryption key** command in the `master` database to create a database encryption key. The syntax is:

```
create encryption key keyname
    [for algorithm]
    for database encryption
    [with
        {[master key]
        [key_length 256]
        [init_vector random]
        [[no] dual_control]}
```

where:

- *keyname* – must be unique in the user's table, view, and procedure name space in the `master` database.
- `for algorithm` – specifies the algorithm. Currently, the only supported algorithm is Advanced Encryption Standard (AES).
- `for database encryption` – explicitly specifies that you are creating an encryption key to encrypt an entire database, rather than a column.
- `master key` – is required for full database encryption. SAP ASE returns an error if the master key does not already exist.
- `key_length 256` – is the size, in bits, of the key you are creating. The only valid length for a database encryption key is 256; SAP ASE returns an error message if you use any other size.
- `init_vector random` – is required for full database encryption. If you specify `init_vector null`, as you can for creating a column encryption key, SAP ASE returns an error.
- `[no] dual control` – indicates whether the database encryption key must be encrypted using dual controls. By default, dual control is not configured.

This example creates a database encryption key that is protected by the master key:

```
sp_configure 'enable encrypted columns', 1
create encryption key master with passwd "testpassword"
```

```
set encryption passwd 'testpassword' for key master
create encryption key dbkey for database encryption
```

See also

- *Changing a Database Encryption Key* on page 67
- *Dropping a Database Encryption Key* on page 68
- *Back Up the Database Encryption Key* on page 78
- *create encryption key* on page 122
- *drop encryption key* on page 152

Changing a Database Encryption Key

To change the manner a database encryption key is protected, as well as who its owner is, use the **alter encryption key** command.

You cannot regenerate a database encryption key for a database.

- To change a database encryption key:
  1. Decrypt the database protected by the database encryption key.
  2. Drop, and re-create the database encryption key.

  Note: You cannot convert a column encryption key into a database encryption key. SAP ASE displays an error message if you alter a different type of encryption key into a database encryption key using the **for database encryption** option.
- To simply change the way a database encryption key is protected, rather than change the database encryption key altogether, use this syntax:

  ```
  alter encryption key key_name
  for database encryption
  modify encryption with {[master key]
      [[no] dual_control}
  ```
- To change the owner of a database encryption key:

  ```
  alter encryption key [[database.][owner].]dek_name
      modify owner user_name
  ```

  The permission to run this option is the same as the permission for **alter encryption key**.

See also

- *Dropping a Database Encryption Key* on page 68
- *Back Up the Database Encryption Key* on page 78
- *create encryption key* on page 122
- *drop encryption key* on page 152
- *Creating the Database Encryption Key* on page 65

Dropping a Database Encryption Key

To drop the database encryption key, use the **drop encryption key** command.

The syntax is:

```
drop encryption key key_name
```

This command deletes the database encryption key from the `sysencryptkeys` table in the `master` database.

Note: This command fails if the database encryption key you are dropping is still being used to encrypt a database.

See also

- *Changing a Database Encryption Key* on page 67
- *Back Up the Database Encryption Key* on page 78
- *create encryption key* on page 122
- *drop encryption key* on page 152
- *Creating the Database Encryption Key* on page 65

Create an Encrypted Database

To create a fully encrypted database, use the **create database** command.

Specify whether to encrypt a database when you create it, and data inserted into the database is automatically encrypted. The size of the database does not change when it is encrypted, and all storage access functions work identically whether a database is encrypted or not. The types of databases that support encryption are:

- Normal user database
- Temporary database
- Archive database

You cannot encrypt an in-memory database.

To create an encrypted database, use:

```
create [temporary] database database_name
    encrypt with key_name
```

Where:

- *database_name* is the name of the encrypted database you are creating.
- *key_name* is the name of the database encryption key.

To create an encrypted archive database, use:

```
create archive database database_name
    encrypt with key_name
```

Where:

- *database_name* is the name of the archive database you are creating
- *key_name* is the same key that you used to encrypt the database that was backed up. SAP ASE verifies that *key_name* matches during the database load. If it does not match, the restoration fails.

Example

Creates an encrypted database called `demodb` with data on device `demodev` and log on device `demologdev`, using an encryption key called `dbkey`:

```
create database demodb on demodev log on demologdev encrypt with
dbkey
```

Usage

There is no special permission to use the `encrypt with` option of the **create database** command. Users however, need **select** permission on the database encryption key to be able to reference it as the *key_name*.

See also

- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *create database for Full Database Encryption* on page 119

Encrypt an Existing Database

In SAP ASE version 16.0, you can encrypt an unencrypted database using the **alter database** command.

Depending on the size of the database, encryption might can take a while. For this reason, the command returns as soon as the database is marked for encryption. Encryption occurs in the background and the process is transparent to users. To check on the status and progress of database encryption, run the **sp_helpdb** system procedure, the **dbencryption_status()** built-in function, or the SAP Control Center user interface. Keep in mind:

- Database encryption occurs while the database is online. This means the database is accessible by other users while it is being encrypted, and does not require you to put it into single-user mode.
- The encryption process does not interrupt any user queries, updates, or insert operations on the database.
- You can suspend and resume database encryption, so that you can resume encrypting the database after restarting SAP ASE.

- The encryption operation is executed page by page.
- You cannot alter archive databases for encryption and decryption.
- SAP ASE records the encryption progress of a database and provides utilities to report its status.

Restrictions:

- You cannot encrypt the `master`, `model`, `dbccdb`, and `dbccalt` databases.
- You cannot decrypt a database that is in the process of being encrypted, or encrypt a database that is being decrypted.
- You cannot unmount a database while it is in the process of being encrypted.
- You cannot load another database on top of a database that is being encrypted.
- Do not execute commands that shrink database size when the database is being encrypted.

The syntax is:

```
alter database database_name
{encrypt with key_name [parallel degree_of_parallelism]
| resume encryption [parallel degree_of_parallelism]
| suspend encryption
}
```

where:

- `encrypt with` *`key_name`* instructs SAP ASE to encrypt the database using *key_name*.

  Specifically, the command retrieves the corresponding key ID from the `sysencryptkeys` system table in the `master` database and set the `encrkeyid` column in its related `sysdatabases` row.

  SAP ASE fails to run **alter database** and displays an error message if the database is already:
  - Encrypted with another key.
  - Being encrypted.

  If you run this command on a partially encrypted database that is not currently being encrypted, SAP ASE treats the command as if you specified the `resume encryption` option, as long as the key name is the same as the previously specified key.
- `parallel` *`degree_of_parallelism`* determines how many worker threads to initiate for the task.

  Create a thread for each database storage virtual device, as long as the number is equal to or fewer than "`number of worker processes`" configuration. The *degree_of_parallelism* number should be no larger than the number of database devices because additional worker threads do not improve encryption performance. If you do not specify *degree_of_parallelism*, SAP ASE internally defines the value based on the number of online engines, as well as how the database is distributed across various devices.
- `resume encryption` resumes the encryption process from the page where encryption was previously suspended.

The command fails if:

- There is an encryption process already running in SAP ASE.
- Encryption was never started on the database.
- The encryption process already completed.

You can use `parallel` *`degree_of_parallelism`* with `resume encrypt`.

- `suspend encryption` terminates all encryption worker threads that are encrypting data. SAP ASE records the progress of encryption so that `resume encryption` can restart encryption where the previous encryption process stopped. SAP ASE ignores this command if there is no encryption in progress.

This example alters an existing database called `existdb` for encryption using an encryption key called `dbkey`:

```
alter database existdb encrypt with dbkey
```

The example does not specify the parallel degree, leaving it up to SAP ASE to determine how many worker threads should be initiated to encrypt `existdb` in parallel.

In addition to the parallel degree, another major factor that affects database encryption performance is the buffer pool size. A sufficient buffer cache and appropriate size of buffer pool enable SAP ASE to load a large chunk of pages into memory for every disk read, perform encryption, and write them back.

The following example shows the steps you can take to configure both the buffer cache and buffer pool size for a database called `demodb` that will be encrypted:

1. Create a specific data cache for `demodb`:

   ```
   sp_cacheconfig demodb_cache, '10M'
   ```

   This creates a named buffer cache called `demodb_cache` with 10MB of space for database pages.
2. Create the specific size of buffer pool . The buffer pool size should be 8 times of database page size. For example, the database page size is 2K by default, therefore the buffer pools size should be 8 x 2 = 16K:

   ```
   sp_poolconfig demodb_cache, '10M' , '16k'
   ```

   This creates a 10MB buffer pool of buffers with a size that is 16K in the named cache called `demodb_cache`.
3. Bind the database to the buffer cache:

   ```
   sp_bindcache demodb_cache, demo_db
   ```

See also

- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *Suspend the Encryption Process* on page 75
- *Resume the Encryption Process* on page 76

- *alter database for Full Database Encryption* on page 113
- *Decrypt an Encrypted Database* on page 76

Encryption Status and Progress

There are two ways to find out what the encryption status is on a database.

To obtain information on whether a database is encrypted or not, as well as how far along the encryption process has gone on a database being encrypted, use:

- The **sp_helpdb** system procedure. The syntax is the following, where *database_name* is the name of the database:

  ```
  sp_helpdb database_name
  ```
- The **dbencryption_status** built-in function. Use status to get information on whether a database is encrypted, and progress to find out how far along the encryption process has gone:
  - `select dbencryption_status("status", db_id("existdb"))`
  - `select dbencryption_status("progress", db_id("existdb"))`

Performance Considerations

When an existing database is being encrypted, it is still kept online. Take performance issues into consideration to mitigate the impact on user access to the database, as well as general SAP ASE response time.

Factors to take into account for good database encryption performance include:

- The number of SAP ASE engines on a multiprocessor machine
- The number of disks the database is stored across
- The buffer pool size associated with the database

Specifying the parallel degree value in **alter database** for encryption or, decryption, essentially tells SAP ASE how many worker threads to initiate when executing the operation. Since worker threads run concurrently, it is better when they are distributed across multiple CPUs. At the same time, it is better to avoid overwhelming CPU resources, since this could reduce the general response time from SAP ASE. For this reason, take the number of SAP ASE engines into consideration when deciding on the parallel degree value.

Device I/O is a major bottleneck during database encryption. SAP ASE can tackle this from two angles:

- If every separate device is assigned a worker thread, device I/O can be carried out independently and concurrently for best throughput. Therefore parallel degree should consider the number of disks the database is stored across.

- Performance will benefit if a big chunk of pages can be processed for every device read/write. The database must be online while the encryption/decryption is in progress. For this reason, instead of allocating a proprietary buffer, existing buffer manager mechanism has to be leveraged to solve synchronization problem. In this respect, you can create sufficient buffer cache and large I/O size of buffer pool to help SAP ASE improve its encryption performance.

This example shows how to configure both the buffer cache and pool size to fully encrypt a database called `demodb`, which has its data and log distributed across 11 devices:

```
> select dbid, segmap, lstart, size, vstart, vdevno from sysusages
where dbid=db_id('demodb')

dbid   segmap lstart     size       vstart      vdevno
------ ------ ---------- ---------- ----------- -----------
   4        3          0      92160           0           1
   4        4      92160      30720           0           2
   4        3     122880     184320       92160           1
   4        4     307200      61440       30720           2
   4        3     368640     419840      276480           1
   4        4     788480      61440       92160           2
   4        3     849920     122880      696320           1
   4        4     972800     153600      153600           2
   4        3    1126400     819200      819200           1
   4        3    1945600    1638400           0           3
   4        3    3584000    1638400           0           4
   4        3    5222400    1638400           0           5
   4        3    6860800    1638400           0           6
   4        3    8499200    1638400           0           7
   4        3   10137600    1638400           0           8
   4        3   11776000    1638400           0           9
   4        3   13414400    1638400           0          10
   4        3   15052800    1638400           0          11
   4        4   16691200     204800      307200           2
```

1. Configure buffer cache and buffer pool size:
   a. Create a specific data cache for `demodb`:

      ```
      sp_cacheconfig demodb_cache, '100M'
      ```

      This creates a buffer cache named `demodb_cache` that has 100MB of space for database pages.
   b. Create the specific size of buffer pool, where the buffer pool size is 8 times the size of the database page size:

      ```
      sp_poolconfig demodb_cache, '100M' , '16k'
      ```

      Since the default database page size is 2K, the buffer pool size should be 8 X 2 = 16KB.
      This creates a 100MB buffer pool with 16K buffers in the named cache `demodb_cache`.
   c. Bind the database to the buffer cache:

      ```
      sp_bindcache demodb_cache, demo_db
      ```

This binds the database `demo_db` to the created buffer cache `demodb_cache`.

2. Determine what parallel degree to use. In this example, there are 8 SAP ASE engines configured:

```
[Thread Pool:syb_default_pool]
```

Number of threads = 8

The maximum number of worker thread should not exceed 8.

In the meantime, with SAP ASE using 11 database devices, it needs, at most, 11 worker threads to perform device I/O in parallel. Since 11 worker threads would strain the eight engines, the parallel degree should be set to 8. However to allow SAP ASE to maintain its response time and perform other operations, avoid occupying all of its CPU resources by selecting a parallel degree of 6.

a. Make sure sufficient worker threads are configured:

```
sp_configure 'number of worker processes', 6
```

b. Alter database `demodb` for encryption:

```
alter database demodb encrypt with dbkey parallel degree 6
```

sp_who shows 6 worker threads:

```
>sp_who
fid    spid    status    loginame    origname
    hostname    blk_spid    dbname
  tempdbname    cmd
        block_xloid        threadpool
------ -------- ---------------- ---------
……
  0     16    sleeping   NULL       NULL     NULL             0
master
   master DB    ENCRYPTION CONTROLLER        0
NULL
  16    1     sleeping   NULL       NULL     NULL             0
master
   master      WORKER PROCESS             0                NULL
  16    17    sleeping   NULL       NULL     NULL             0
master
   master     WORKER PROCESS              0                NULL
……
```

sp_helpdb can report the encryption progress and status:

```
1> sp_helpdb demodb
2> go
name     db_size    owner  dbid  created       durability
lobcomplvl inrowlen status
-------- ---------- ------ ----- ------------- ----------
---------- -------- ----------------
demodb   33000.0 MB sa      4   Sept 27, 2013 full             0
NULL     encryption in progress: 18%
```

You can also use the **dbencryption_status** function to get encryption status and progress:

```
1> select dbencryption_status("status", db_id('demodb'))
2> go
-----------
2
1> select dbencryption_status("progress", db_id('demodb'))
2> go
-----------
21
```

This shows that 21 percent of database pages has been encrypted.

You can also use **dbencryption_status** to find the progress on a specific fragment:

```
1> select dbencryption_status("progress", db_id('demodb'),
92160)
2> go
-----------
83
```

This shows that 83 percent of pages in the fragment with a logical page start of 92160 has been encrypted.

Encrypted databases consume more buffers for encryption and decryption than nonencrypted databases. If clean buffers are unavailable because of encryption and decryption:

- Increase the buffer pool size and the buffer pool wash
- Configure **housekeeper free write percent** to a value that allows the housekeeper task to wash buffers more frequently

Suspend the Encryption Process

To stop encrypting a database in the process of being encrypted, use the the `suspend encrypt` option of the **alter database** command:

```
alter database database_name
    suspend encryption
```

See also

- *Encrypt an Existing Database* on page 69
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *Resume the Encryption Process* on page 76
- *alter database for Full Database Encryption* on page 113

The quiesce database Command and Fully Encrypted Databases

When you run the **quiesce database** command on a database that is being encrypted, SAP ASE puts the encryption process on hold.

You need not run the `suspend encryption` option of **alter database** after you run **quiesce database**; **quiesce database** automatically suspends the I/O operation on the database.

After the quiesce mode is released, the task of encrypting (or decrypting) resumes automatically; you need not run the **resume encryption** option in the **alter database**.

Resume the Encryption Process

To resume encrypting a database that had its encryption process interrupted or suspended, use the `resume encryption` option of the **alter database** command:

```
alter database database_name
    resume encryption [parallel degree_of_parallelism]
```

See also

- *Encrypt an Existing Database* on page 69
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *Suspend the Encryption Process* on page 75
- *alter database for Full Database Encryption* on page 113

Decrypt an Encrypted Database

To decrypt a fully encrypted database, use the **alter database** command.

The syntax is:

```
alter database database_name
  {decrypt [with key_name] [parallel degree_of_parallelism]
| resume decryption [parallel degree_of_parallelism]
| suspend decryption}
```

where:

- *database_name* – is the name of the fully encrypted database you want to decrypt.
- *key_name* – (optional) is the same database encryption key you used to encrypt the database. If you specify a different key name, the command fails and SAP ASE displays an error message.

- `resume decryption` – resumes the decryption process for the database in which an earlier decryption process has been suspended. SAP ASE ignores this command if the *database_name* is already completely decrypted.
- `parallel` *degree_of_parallelism* – specifies how many worker threads to initiate for the task.
- `suspend decryption` – terminates the decryption process. SAP ASE records where the process was stopped, so that `resume decrypt` can restart the decryption process at the correct place in the database.

You must have **select** permission on the database's *key_name* to use this command.

Recover Fully Encrypted Databases

If SAP ASE cannot retrieve the database encryption key during start-up because the master or dual master key is unavailable, SAP ASE ignores the encrypted database.

SAP ASE needs access to the database encryption key to know what to do with fully encrypted databases. The database encryption key itself is also encrypted, and is decrypted by the master key.

To connect to the server after you restart SAP ASE, the password holder for the master or dual master key can set the encryption password:

```
set encryption passwd for key [dual] master
```

This allows the master key to decrypt the database encryption key, at which point the database encryption key can bring the fully encrypted database online:

```
online database encrypted_database_name
```

Database recovery then occurs as the server comes back online.

You can also set up an automatic recovery; see *Starting Adaptive Server in Unattended Start-Up Mode* in the *Encrypted Columns Users Guide*.

Back Up (Dump) a Fully Encrypted Database

Backing up a fully encrypted database is the same as normal, unencrypted databases, since the encryption process is performed transparently.

To load a back-up dump of an encrypted database, it must use the same encryption key that was used to encrypt the dump.

The database encryption key is stored in the `master` database, outside of the database you are backing up. For this reason, the backup process is not automatically applied to the database encryption key when you execute the **dump database** command; you must independently back up the database encryption key and the master key separately from the database backup.

To back up the key values, either:

- Use the **ddlgen** utility to generate a DDL statement, or;
- Back them up directly.

Back Up the Database Encryption Key

To resume recoverability, you must back up the database encryption key, the master or dual master key, and the encrypted database.

This example uses the **ddlgen** utility to generate SQL statements on database encryption keys:

```
ddlgen -Usa -P -Sserver -TEK -Nmaster.owner.dek_name
```

The syntax is similar when generating SQL statements for the [dual] master key.

See also

- *Changing a Database Encryption Key* on page 67
- *Dropping a Database Encryption Key* on page 68
- *create encryption key* on page 122
- *drop encryption key* on page 152
- *Creating the Database Encryption Key* on page 65

Restore (Load) Backups of Fully Encrypted Databases

Restore a fully encrypted database as you would a normal, unencrypted database.

Before you can load an encrypted database dump:

1. Restore the master key and database encryption key.
2. Create the target database for encryption using the same database encryption key you used for the database you are loading.

Use this command to restore your encrypted database, where *database_name* is the name of the encrypted database you are restoring:

```
load database database_name
```

Note: You cannot use the verification option (`load database database_name with verify only = full`) with encrypted databases. When you specify this option, Backup Server reads all rows and checks that the row formats are valid. Since Backup Server cannot understand encrypted text, the command fails and Backup Server displays an error message.

When you perform **load database** to restore an encrypted database, SAP ASE verifies that the taget database:

- Is an encrypted database. If it is not, SAP ASE displays an error message and the **load database** command fails.

- Has the correct database encryption key. If the database encryption key does not match, SAP ASE displays an error message.

Loading Behavior of Encrypted Databases

Loading behavior differs, depending on the encryption status of both the target database and the database or transaction log being restored.

Loading Behavior	Unencrypted Target Database	Encrypted Target Database	Partially Encrypted Target Database	Partially Decrypted Target Database
Unencrypted Database Dump	Allowed.	Allowed only if using the `with override` clause. Dump security status is reflected in target database.	Allowed only if using the `with override` clause. Dump status is reflected in target database.	Allowed only if using the `with override` clause. Dump security status is reflected.
Unencrypted Transaction Dump	Allowed.	Allowed. Marks the target database as partially encrypted.	Allowed. The target database retains its status as partially encrypted.	Allowed. The target database retains its status as partially encrypted.
Encrypted Database Dump	Not allowed.	Allowed.	Allowed. Dump security status is reflected in the target database.	Allowed only if using the `with override` clause. Dump security status is reflected in the target database.
Encrypted Transaction Dump	Not allowed.	Allowed.	Allowed. The target database retains its status as partially encrypted.	Not allowed.
Partially Encrypted Database Dump	Not allowed.	Allowed. Dump security status is reflected in the target database.	Allowed. The target database retains its status as partially encrypted.	Allowed only if using the `with override` clause. Dump status is reflected in the target database.

Loading Behavior	Unencrypted Target Database	Encrypted Target Database	Partially Encrypted Target Database	Partially Decrypted Target Database
Partially Encrypted Transaction Dump	Not allowed.	Allowed. Dump security status is reflected in the target database.	Allowed. The target database retains its status as partially encrypted.	Not allowed.
Partially Decrypted Database Dump	Not allowed.	Allowed only if using the `with override` clause. Dump status is reflected in the target database.	Allowed only if using the `with override` clause. Dump security status is reflected in the target database.	Allowed. The target database retains its status as partially decrypted.
Partially Decrypted Transaction Dump	Not allowed.	Not allowed.	Not allowed.	Allowed. The target database retains its status as partially decrypted.

Dropping a Database That is Being Encrypted

You can drop a database that is being encrypted or decrypted.

When you execute the **drop database** command on a database that is being encrypted or decrypted, **drop database** terminates the encryption/decryption process, searches the `sysattributes` system table, cleans up all the progress information, and then drops the database.

Mounting and Unmounting a Fully Encrypted Database

You cannot mount a database that is being encrypted or decrypted.

You cannot mount an encrypted database.

Do not use the **umount database** command on an encrypted database; the command fails and SAP ASE displays a message similar to:

```
Could not unmount encrypted database 'mydatabase'.
```

To unmount an encrypted database, decrypt it first.

See also

- *Decrypt an Encrypted Database* on page 76

Archive Databases and Full Encryption

Archive databases are read-only. The encryption syntax indicates that an archive database can load an encrypted database dump.

As with database backups and loads, restore the master key and database encryption key, and associate the DEK with the archive database.

To dump or load a fully encrypted archive database, perform the same steps as with normal databases.

To create an archive database, use:

```
create archive database database_name
    encrypt with key_name
```

where:

- *database_name* is the name of the archive database you are creating
- *key_name* is the same key that you used to encrypt the database that was backed up (dumped). SAP ASE verifies that *key_name* matches during the database dump. If it does not match, the restoration fails.

Unlike normal databases, an archive database provides a modified page section that stores page modification or allocation information due to redos/undos and transaction loading operations. When you encrypt an archive database, encrypt the data in the modified page section as well, using the database encryption key from the archive database.

There is no special permission to use the `encrypt with` option of the **create archive database** command. Users however, need **select** permission on the database encryption key to reference it as the *key_name*.

See also

- *Back Up (Dump) a Fully Encrypted Database* on page 77
- *Restore (Load) Backups of Fully Encrypted Databases* on page 78

Full Database Encryption and System Changes

There are system changes associated with the full database encryption feature of SAP ASE 16.0.

- **alter database** command
- **alter encryption key** command
- **create archive database** command

- **create database** command
- **create encryption key** command
- **dbencryption_status()** built-in function
- **ddlgen** utility
- **sp_encryption** system procedure
- **sybmigrate** utility
- `sysdatabases` system table

See also

- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

Scalability Enhancements and Features

SAP ASE version 16.0 improves scalability by improving run-time logging and lock, metadata, and latch management.

Run-Time Logging Enhancements

SAP ASE 16.0 includes run-time logging enhancements.

Transferring log records to `syslogs` requires that SAP ASE acquire a log lock, which can become a point of contention in a busy OLTP system. SAP ASE reduces contention and improves run-time logging performance by using a user log cache for each transaction. The user log cache buffers log records before they are transferred to `syslogs`, and allows SAP ASE to send batches of log records to syslogs instead of individual log records.

To take full advantage of the user log cache, SAP ASE perferably transfers its log records to the log only once when the transaction completes. However, when concurrent open transactions make changes to the same data page in a busy OLTP system that uses datarow-locked tables, SAP ASE may need to transfer the log records from a user log cache to `syslogs` before the transaction completes, which decreases the amount of batching and increases contention on the log lock. In a busy OLTP system that is updating the same table with many concurrent transactions, the benefit of caching the log records in the user log cache may be entirely negated when SAP ASE frequently transfers the log records from the user log cache to the log before the transaction completes.

To reduce contention, SAP ASE version 16.0 and later partitions each user log cache into a number of smaller blocks, each the size of the server's logical page size. SAP ASE version 16.0 adds log records directly to the current active block within the user log cache (the same manner that earlier releases add log records directly to the user log cache). When concurrent open transactions make changes to the same data page, instead of moving log records directly from the user log cache to `syslogs`, SAP ASE adds (or links) the current block within the user log cache to the end of the global queue; it can later transfer the log records from the global queue to `syslogs`. This enables SAP ASE to efficiently batch additions to `syslogs`, improving the performance of run-time logging, regardless of issues associated with datarow-locked tables.

After the current block is added to the global queue, a new free block within the user log cache becomes the current block, and continues to accept log records from the transaction.

Use the **user log cache queue size** configuration parameter to enable and disable this functionality. When **user log cache queue size** is set to:

- 1 – SAP ASE uses the queuing strategy (the default)

- 0 – SAP ASE use the pre-16.0 version behavior

You must set **user log cache size** to be at least four times the size of the server's logical page size if you configure SAP ASE to use the queuing strategy. That is, each user log cache must have at least 4 user log cache blocks (each block is the size of the logical page size). If the value is DEFAULT, SAP ASE version 16.0 sets the value to four times the size of the server's logical page size

Keep the following in mind when setting the value for **user log cache size** and **user log cache queue size**:

- You cannot set **user log cache size** to a value that is less than 4 times the logical page size if **user log cache queue size** is set to a value greater than 0.
- You cannot change the value of **user log cache queue size** from 0 to 1 if **user log cache size** is set to a value less than or equal to 4 times the logical page size.

Lock Management Enhancements

SAP ASE version 16.0 and later includes several enhancements to lock management.

Spinlocks in versions of SAP ASE earlier than 16.0 occasionally result in lock contention. SAP ASE version 16.0 reduces lock contention by optimizing the following areas:

- Engine lock transfer – Improves the transfers of locks between global pool and engine local caches.
- Engine lock caching – Optimizes the number of locks an engine can cache locally by:
  - Increasing the default engine local cache size
  - Increasing the transfer size of locks from global to local caches.
  - Improves lock transfers between local and global by using of blocks of locks instead of individual locks
  - Draining or reclaiming locks from local to global caches less frequently
- `LOCK_VERIFY` operations – Optimizes `LOCK_VERIFY` operations to deploy an optimistic approach when possible and avoid taking locks.
- Lock promotion - Tracks repeatedly failed lock promotion attempts, and disables lock promotion for the DML statement incurring the failed lock promotion after a number of attempts.
- Log semaphore locks - Avoids taking locks for log semaphores since only one lock can be held by the `semawait`.
- Deadlock checking - Ensures that specifying a small value for the **deadlock checking period** never results in a value less than 1, and that deadlock checks are performed on a dedicated service thread
- Hot DOL tables - Allows DOL tables to be hot tables, which alleviates high table lock contention

These features are intended to enhance performance: you should not notice any performance degradation when SAP ASE uses these enhancements. The lock management enhancements

are enabled by default, and you need not to perform any configuration to enable these enhancements.

Metadata and Latch Management Enhancements

SAP ASE Enterprise version 16.0 and later decreases CPU utilization for latch conflicts in environments that have very high transaction rates.

Including, decreased contention:

- On internal structures during very high transaction rate cross-database queries and transactions, including contention on locks, latches, and data caches due to implicit or explicit cross-database references
- During high transaction-rate operations on tables with a large number of partitions
- On internal structures during very high rates of create and drop table, including decreased lock and latch contention on some system tables

These features are intended to enhance performance: you should not notice any performance degradation when SAP ASE uses these enhancements. The metadata and latch management enhancements are enabled by default, and you need not to perform any configuration to enable these enhancements.

SAP ASE version 16.0 and later decreases CPU utilization during latch conflicts in very high transaction rate environments.

Concurrent operations on a single page can cause contention during periods of times with extremely high transaction rates, manifesting as data cache spinlock contention on a single cache partition. Typically, you can trace this contention to a large number of latch requests on a single data page. The **sp_chgattribute exp_row_size** parameter helps manage space on the server. SAP ASE reserves space on data pages according to the size to which you set **exp_row_size**, which can limit the number of rows per page.

SAP ASE versions 16.0 and later allows you to set the values for **exp_row_size** for tables with only fixed-length columns. To decrease contention, decrease the number of rows per page by setting **exp_row_size** to the logical page size, (for example, 8,192).

High workload scenarios can increase contention on procedure cache, which SAP ASE alleviates by setting aside local memory for each engine. Because this memory is local, accessing it does not cause contention.

In versions earlier than 16.0, SAP ASE set aside 25% of the procedure cache for the engine local cache (ELC) for use by requests of a specific size. Using trace flag 758 increased the size to 50% of the procedure cache, and made the ELC available for all request sizes.

SAP ASE versions 16.0 and later enable ELC by default. 50% of the procedure cache is used for ELC, which services all request sizes. The **engine local cache percent** configuration parameter determines the size of the ELC. The default value, 50, means that the local cache uses fifty percent of the procedure cache, and each engine receives:

```
((0.50 * procedure_cache_size) / number_of_online_engines)
```

Increasing the size of ELC can decrease contention on the procedure cache. For example, to increase ELC from default 50% to 60%, issue:

```
sp_configure 'engine local cache percent', 60
```

Additionally, SAP ASE versions 16.0 and later include these configuration parameters to control the engine local cache:

- **enable large chunk elc** (replaces traceflag 758)
- **large allocation auto tune** (replaces traceflag 753)

These configuration parameters are enabled by default.

Monitoring Threshold-Based Events

SAP ASE versions 16.0 and later include the ability to configure, record, and list threshold events.

To configure the amount of time a query spends using CPU, use the **sp_add_resource_limit cpu_time** limit type.

To record events to the `monThresholdEvent` table, set the **sp_add_resource_limit** *action* parameter to 5 (see the *Reference Manual: Procedures*).

The syntax is:

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

In this example, SAP ASE issues a warning when the CPU time of the query batch takes longer than 120 seconds to execute, and records the events to `monThresholdEvent`:

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,cpu_time, 120, 2,
5, 2
```

`monThresholdEvent` lets you record all violations for configured resource limitations, and allows you to determine the thresholds for:

- `tempdb` usage per transaction
- Total runtime of a query or transaction
- Number of rows fetched or written by a query
- Number of rows read by a query
- Estimated plan cost of a query
- Total logical/physical I/Os of a query
- Total CPU time of a query

Multiple Triggers

SAP ASE version 16.0 introduces the ability to create multiple triggers, as well as to specify the order in which the triggers are fired after statement execution.

Creating Multiple Triggers

Create up to 50 different triggers on a table for each command (**insert**, **update**, and **delete**). Use the new **order** parameter in the **create trigger** command, and specify the order that the triggers fire. You can create multiple triggers without an order clause.

The syntax is:

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    for {insert | update | delete}
    [order integer]
    as sql_statements
```

`order` *`integer`* specifies a partial or full ordering of trigger firing:

- Full ordering occurs when you create all the triggers using the `order` clause.
- Partial ordering occurs if you do not specify the `order` clause on some of the triggers. Triggers without the `order` clause implicitly take order number 0 and do not have a defined order, except that they fire after those triggers created using `order`.

Note: You can only use the `order` *`integer`* clause with `for {insert | update | delete}`; you cannot use it with `instead of {insert | update | delete}` triggers.

If you use a duplicate number for `order`, SAP ASE reports an error. `order` numbers need not be consecutive; in fact, nonconsecutive numbers might be preferable, as they allow you to insert new triggers into the middle of an order.

Use **sp_helptrigger** to list:

- All triggers created on the table specified by *tablename*
- The **insert**, **update**, or **delete** command that provided the triggering action
- The trigger's order number

See also

- *create trigger for Multiple Triggers* on page 144

Changing the Order of When a Trigger is Fired

To change the order in which a trigger is fired, use the **or replace** option in the **create trigger** command, using the same trigger name, action, and trigger body as the original trigger, but specifying a new **order** number.

Order of Triggers in Merge Statements

The `merge` statement fires triggers in a specific order.

The **merge** statement fires triggers on the target table in this order:

1. **insert**
2. **update**
3. **delete**

This means that even if the order number for an **update** statement is lower than the order number for an **insert**, the trigger for the **insert** statement fires first.

Multiple triggers for the same operation, however, are ordered. That is, all the triggers for **insert** are fired in order number first, followed by all the triggers, in order, for **update**, and so on.

In this example, the dbo creates these triggers on the `GlobalSales` table:

- trigger1 for delete with order 1
- trigger2 for delete with order 4
- trigger3 for insert with order 1
- trigger4 for insert with order 5

A merge statement merges data into `GlobalSales`. If the merge includes both insert and delete operations on `GlobalSales`, SAP ASE fires the triggers in this order after execution:

- trigger3
- trigger4
- trigger1
- trigger2

Transactional Behavior with Multiple Triggers

A rollback transaction executed in a trigger causes rollback of the **insert**, **update**, or **delete** statement that fired the trigger, along with any work performed by the trigger.

For multiple triggers, a rollback transaction from one trigger also rolls back the work of other triggers already fired and withholds firing any remaining triggers for the current insert, update or delete command.

Disabling and Reenabling Triggers

You can disable and reenable multiple triggers using the `alter table` command.

Use the **alter table** command to disable or re-enable multiple triggers individually:

To disable or re-enable the trigger, use:

```
alter table table_name {disable | enable} trigger trigger_name
```

@@trigger_name Global Variable

The @@trigger_name global variable returns the name of the trigger that is currently executing..

You can place the following in the body of a trigger or in the body of a stored procedure that is called (at any level of nesting) from a trigger:

```
select @@trigger_name
```

If nested triggers are fired, `@@trigger_name` holds the name of the most recently fired trigger.

Residual Data Removal

Remove residual data to strengthen the security of database data.

Some database operations that delete space do not always physically erase the data. This can pose a security threat, as this residual data may be visible to a user using the **dbcc** utility. SAP ASE version 16.0 introduces a feature that, when enabled, can automatically zero out residual data when users perform these database operations.

Versions of SAP ASE earlier than 16.0 did not physically delete residual data by zeroing out disk pages, which left potentially sensitive data on the disk. In version 16.0 and later, you can mark data as sensitive, and configure SAP ASE to erase residual data after performing delete or update operations.

Note: You can automatically zero out residual data when users performs such database operations.You cannot remove residual data from system databases such as `master`, `sybsystemdb`, and `sybsystemprocs`, or from system tables present in each database.

For information about operations that results in residual data and how to enable the removal of residual data, see *Residual Data Removal* in the *Security Administration Guide*.

Configuration History Tracking

SAP ASE version 16.0 adds the ability to track the history of configuration changes made to your server.

The **sp_confighistory** system procedure manages the history of configuration changes, and stores data about the changes in the `sybsecurity` database.

Configuration properties that are tracked include

- Server-wide configuration parameters
- Database options
- Data cache and data cache pool properties
- Engine threads
- Changes to the server configuration file.

You must install the the `sybsecurity` auditing database to track these properties.

sp_confighistory displays SAP ASE configuration changes, including which configuration option has been changed, the old and new values, the user who made the change, and when the change was made. SAP ASE stores records of configuration changes in the `sybsecurity` database. Query the `ch_events` view or run **sp_confighistory** to access these records.

For example, the output below includes changes that include enabling auditing and changing the value of **max online engines** from 5 to 7:

```
area type target element oldvalue newvalue mode timestamp username
instanceid
---- ---- ------ ------- -------- -------- ---- --------- --------
----------
AUDIT global auditing  NULL NULL off on NULL Jul 15 2013 2:22PM sa
NULL
SERVER sp_configure NULL max online engines 5 7 static Jul 15 2013
2:23PM sa NULL
```

Configuring SAP ASE to Track Configuration Changes

To install **sp_confighistory**, run the `installsecurity` script.

Prerequisites

Note: `ch_events` collects information from all the audit tables, and becomes out of sync if you add or remove audit tables. If this occurs, `ch_events` may not include some configuration history record changes, or you may see error messages 208 (`table not found`) and 4413 (`view unusable`) when you query `ch_events`.

Use **sp_confighistory create_view** to update `ch_events` when you add or remove audit tables. **sp_confighistory create_view** drops the view if it exists, and creates a new view that corresponds to the current audit tables.

Install and enable the audit system. See the *Security Administration Guide*.

Task

1. Enable configuration history tracking (requires the sa_role, sso_role, or `manage auditing` if granular permission is enabled):

   ```
   sp_audit "config_history", "all", "all", "on"
   ```

   Note: Issuing **sp_audit** is recorded in the configuration history.
2. Enable auditing:

   ```
   sp_configure 'auditing', 1
   ```
3. Move to the `sybsecurity` database:

   ```
   use sybsecurity
   ```
4. Create the `ch_events` view:

   ```
   sp_confighistory create_view
   ```

Changes Captured

When configuration history auditing is enabled, SAP ASE captures a number of events.

The `ch_events` view does not record changes if the new value is the same as the old value.

`ch_events` records these configuration changes (described in detail below):

- Changes to the configuration File
- Reading the configuration File
- **sp_configure** Changes
- Changes to server options
- Changes to database options
- Changes to cache configuration
- Changes to trace flags and switches
- Changes to number of engines
- SAP ASE startup and shutdown events
- Enabling or disabling auditing

Startup Configuration Changes

If you modify the SAP ASE configuration file while the SAP ASE is shut down, SAP ASE records any changes to the configuration in the ch_events table it starts (recording a value of NULL for the `mode` and `username` values for these changes).

Reading the Configuration File

`ch_events` records the event when you read, write, verify, and restore the configuration file, but does not record the configuration value changes. For example, if you change the value for **number of user connections** and then issue:

```
sp_configure "configuration file", 0, "read", "srv.config"
```

`ch_events` records that you read the configuration file, but does not record the configuration value changes.

sp_configure Changes

SAP ASE records all changes made by **sp_configure**, including:

- Name of the configuration parameter
- Old configuration value
- New configuration value
- Whether the parameter is dynamic or static
- Timestamp of the date and time the change was made
- Login of the user making the change

Configuration changes caused by reading from the configuration file are not recorded. That is, SAP ASE records the reading, writing, verifying, and restoring operations, but does not record the configuration changes caused by a reading operation. You can also change configuration values by reading a different or manually modified configuration file. Although SAP ASE records that it read the file, it does not record the individual parameter changes.

Changes to Server Options

`ch_events` records all changes made by **sp_serveroption**, including:

- Name of affected server
- Name of option that was changed
- Old option value
- New option value
- Timestamp of the date and time the change was made
- Login of the user making the change

Changes to Database Options

`ch_events` records these changes made by **sp_dboption**:

- Name of affected database
- Name of option that was changed
- Old option value
- New option value
- Timestamp of the date and time the change was made

- Login of the user making the change

Changes to Cache Configuration

`ch_events` records all changes made by **sp_cacheconfig** and **sp_poolconfig** to cache configurations.

Recorded changes from **sp_cacheconfig** include:

- Name of affected cache
- Old cache size
- New cache size
- Attribute (cache type, cache replacement policy, partition number), if applicable
- Timestamp of the date and time the change was made
- Login of the user making the change
- (Cluster Edition only) Instance to which this change applies

Recorded changes from **sp_poolconfig** include:

- Name of affected cache
- Configuration pool
- Old cache pool size
- New cache pool size
- Name of changed attribute (affected pool, wash size, asynchronous prefetch (APF) percentage)
- Timestamp of the date and time the change was made
- Login of the user making the change
- (Cluster Edition only) Instance to which this change applies

Changes to Trace flags and Switches

`ch_events` records changes to **dbcc traceon | off** and **set switch on | off**.

Recorded changes from **dbcc traceon | off** include:

- Trace flag affected
- Session ID
- Old trace flag state (**on** or **off**)
- New trace flag state (**on** or **off**)
- Timestamp of the date and time the change was made
- Login of the user making the change
- (Cluster Edition only) Instance to which this change applies

Recorded changes from **set switch on | off** include:

- Switch number affected
- Old switch state (**on** or **off**)

- New switch state (**on** or **off**)
- Name of changed attribute (server-wide or session-specific, with override, with no_info)
- Timestamp of the date and time the change was made
- Login of the user making the change

Changes to Number of Engines

The changes tracked by **create thread pool**, **alter thread pool**, and **drop thread pool** include:

- Name of pool name affected
- Old pool size
- New pool size
- Name of changed attribute (new pool name, idle timeout, and so on)
- Timestamp of the date and time the change was made
- Login of the user making the change
- (Cluster Edition only) Instance to which this change applies

SAP ASE Startup and Shutdown Events

`ch_events` records this information for startup, **shutdown**, **shutdown with nowait**, and abrupt (unscheduled) shutdown events for SAP ASE and instances from the Cluster Edition:

- Name of the action (startup, **shutdown**, **shutdown with nowait**, abrupt shutdown).
- Time spent waiting for a shutdown. Not applicable for **shutdown with no_wait**.
- Name of the host on which the server starts.
- Timestamp of the date and time the change was made.
- Login of the user making the change.
- (Cluster Edition only) Instance to which this change applies.

Note: Because `ch_events` records the shutdown when you issue the **shutdown** command, `ch_events` may record multiple shutdowns during a polite shutdown if you issue the command more than once.

Enabling or Disabling Auditing

`ch_events` records this information about tracking, global auditing, and configuration history auditing:

- Name of the action (**enable** or **disable**)
- Timestamp of the date and time the change was made
- Login of the user making the change

Querying ch_events to View Changes

SAP ASE includes the `ch_events` view as part of the `sybsecurity` database. .

`ch_events` present the configuration change history data in an easy to read format. You can query `ch_events` directly, or use the **sp_confighistory** system procedure to generate reports on configuration changes history. Either method provides the same information.

Using the **select** command provides the flexibility of the Transact-SQL™ language to qualify your result set (you must first move to the `sybsecurity` database before selecting from the `ch_events` view). **sp_confighistory** provides a more streamlined result set.

For example, if you make these configuration changes in SAP ASE:

```
sp_dboption sybsystemprocs, "delayed commit", false
sp_cacheconfig pub_cache, '10M'
sp_cacheconfig pub_log_cache, '2000K', logonly
```

Then shut down and restart the server, **sp_confighistory** returns:

```
sp_confighistory

 area     type            target         element            oldvalue
        newvalue    mode timestamp           username instanceid
 -------- --------------- -------------- -------------------
--------
        ----------- ---- ------------------- -------- ----------
 AUDIT    global auditing NULL           NULL                off
        on          NULL Aug 22 2013 11:56AM sa               NULL
 DATABASE sp_dboption     sybsystemprocs delayed commit      true
        false       NULL Aug 22 2013  3:16PM sa               NULL
 CACHE    sp_cacheconfig  pub_cache      NULL                10240
        not changed NULL Aug 22 2013  3:18PM sa               NULL
 CACHE    sp_cacheconfig  pub_log_cache  cache type: logonly 2000
        not changed NULL Aug 22 2013  3:19PM sa               NULL
 SUSD     shutdown        NULL           NULL                NULL
        NULL        NULL Aug 22 2013  3:49PM sa               NULL
 SUSD     startup         NULL           tigger              NULL
        NULL        NULL Aug 22 2013  3:50PM NULL           NULL
```

Include the date with **sp_confighistory** to select the changes over a period of time. This example shows all changes made after August 23, 2013:

```
sp_confighistory "Aug 23 2013"

 area type     target element oldvalue newvalue mode timestamp
        username instanceid
 ---- -------- ------ ------- -------- -------- ---- ---------
-------------------
        -------- ----------
 SUSD shutdown NULL   NULL    NULL     NULL     NULL Aug 23 2013
9:00AM
        sa             NULL
```

```
 SUSD startup  NULL   tigger  NULL     NULL     NULL Aug 23 2013
10:38AM
         NULL           NULL
```

Issuing **select** provides this result set:

```
use sybsecurity
go
select * from ch_events
go

area        type                target           element
oldvalue    newvalue     mode
                  timestamp               username       instanceid
----------  ------------------  ----------------
-----------------    ----------  ---------    --------------
                    ----------------------   -----------
---------------
 AUDIT      global auditing      NULL             NULL
off          on           NULL
                    Aug 22 2013 11:56AM       sa               NULL
 DATABASE   sp_dboption         sybsystemprocs    delayed commit
true         false        NULL
                    Aug 22 2013  3:16PM       sa               NULL
 CACHE      sp_cacheconfig       pub_cache         NULL
10240        not changed  NULL
                    Aug 22 2013  3:18PM       sa               NULL
 CACHE     sp_cacheconfig      pub_log_cache    cache type: logonly
2000         not changed  NULL
                    Aug 22 2013  3:19PM       sa               NULL
 SUSD       shutdown             NULL             NULL
NULL         NULL         NULL
                    Aug 22 2013  3:49PM       sa               NULL
 SUSD       startup              NULL             tiger
NULL         NULL         NULL
                    Aug 22 2013  3:50PM       NULL             NULL
```

Include the **last** parameter to see the last items changed:

```
sp_confighistory last

 area type    target element oldvalue newvalue mode timestamp
         username instanceid
 ---- ------- ------ ------- -------- -------- ---- ----------
-------------------
         -------- ----------
 SUSD startup NULL   tigger  NULL     NULL     NULL Aug 22 2013  3:50PM
         NULL           NULL
```

Cyclic Redundancy Checks for dump database

SAP ASE adds a cyclic redundancy check for accidental changes to raw data for database or transaction dumps created with compression to check and for verification that the compression blocks can be correctly read and decompressed.

The syntax is:

```
dump database database_name to dump_device with
compression=n,verify={crc | read_after_write}
load database database_name from dump_device with verify[only]=crc
```

Where:

- **verify=crc** – indicates that you are performing a cyclic redundancy check.
- **verify=read_after_write** – Backup Server rereads every compressed block after writing and decompresses it. If Backup Server finds an error, it prints the offset in the file in which it finds the error. **verify=read_after_write** is only applicable with the **dump database** command.

This example verifies database `new_dump` before dumping it to the `mydumpdev` device:

```
dump database new_dump to mydumpdev with
compression=x,verify=read_after_write
```

This example performs a cyclic redundancy check as it loads the `new_dump` database dump:

```
load database new_dump from mydumpdev with verify[only]=crc
```

This example performs a cyclic redundancy check and rereads every compressed block before dumping database `new_dump` to the `mydumpdev` device:

```
dump database new_dump to mydumpdev with
compression=x,verify=read_after_write,verify=crc
```

Usage:

- Dumps created without the **verify=crc** parameter use the same format as earlier versions of Backup Server.
- SAP ASE ignores the **verify=crc** option if the database was not originally dumped using **verify=crc**.
- You cannot load dumps that include cyclic redundancy checks with versions of Backup Server that do not include this functionality.
- **verify={crc | read_after_write}** checks are applicable only for files created using the **with compression** parameter.

- **verify=crc** works with any file type, including raw devices, disk files, tapes, pipes, or APIs. However, **verify=read_after_write** requires a 'seek back' for rereading the block, and is applicable only with raw devices and disk files.
- SAP ASE ignores, and does not raise an error message, if you include **verify={crc | read_after_write}** parameters that are not applicable.

Calculating the Transaction Log Growth Rate

SAP ASE version 16.0 adds the ability to calculate the transaction log growth rate for a specified time period.

sp_logging_rate displays the minumum, maximum, and average rate of transaction log growth, in gigabytes per hour, for the period of time you run the system procedure, providing the result as an averaged sum of the calculations, or as iterative results.

This example displays a summary the log growth for the transaction log over a 24 hour period, calculating the growth in one-hour intervals:

```
sp_logging_rate 'sum', '1,00:00:00', '01:00:00'
=========================
Total Summary Information
=========================
Transaction Log Growth Rate      Min GB/h        Max GB/h        Avg
GB/h
---------------------------   --------------  --------------
--------------
                                  0.000000        1.970084
1.566053
```

System Changes

SAP ASE 16.0 adds changes to configuration parameters, commands, system procedures, functions, monitoring tables, and utilities.

Configuration Parameters

SAP ASE version 16.0 introduces changes to configuration parameters.

New Configuration Parameters

SAP ASE 16.0 includes new configuration parameters.

enable utility lvl 0 scan wait

Table 1. Summary Information

Default value	0
Range of values	0 (off), 1(on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Application Functionality

Allows you to run **alter table ... add | drop partition** commands while Adaptive Server runs isolation level 0 scans.

Note: The default value for **enable utility lvl 0 scan wait** depends on the value to which **enable functionality group** is set. If you set enable functionality group to:

- 0 – the default value for **enable utility lvl 0 scan wait** is 0
- 1 – the default value for **enable utility lvl 0 scan wait** is 1

Note: However, if you set **enable utility lvl 0 scan wait** to 1, it uses a value of 1 regardless of what you set **enable functionality group** to. See **enable functionality group**.

(UNIX Only) max network peek depth

Table 2. Summary Information

Default value	0
Range of values	0 – 2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Network Communication

Specifies how many levels deep SAP ASE peeks into a connections operating system receive buffer for a pending cancel. For example, if a client sends a new command followed by a cancel before SAP ASE finishes processing the current command, SAP ASE peeks into the operating system's receive buffer to the depth specified by **max network peek depth**. If the cancel occurs within the specified depth, both the current command and the command preceding the cancel are discarded, and SAP ASE waits for the next command.

aggressive task stealing

Table 3. Summary Information

Default value	1 (on)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	SQL Server Administration

Enabling **aggressive task stealing** sets the SAP ASE scheduler task stealing policy to aggressive.

enable large chunk elc

Table 4. Summary Information

Default value	1 (on)
Range of values	0 (off), 1 (on)

Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	Meta-Data Caches

Enables large allocation in the engine local cache.

engine local cache percent

Table 5. Summary Information

Default value	50
Range of values	0–100
Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	Meta-Data Caches

Allows you to modify the engine local cache as a percentage of procedure cache.

large allocation auto tune

Table 6. Summary Information

Default value	1 (on)
Range of values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	Meta-Data Caches

Configures SAP ASE preallocate large amounts of memory for query execution, which reduces procedure cache contention.

threshold event max messages

Table 7. Summary Information

Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Memory User, Monitoring

Determines the number of events SAP ASE stores in the `monThresholdEvent` table. Once the number of events in the `monThresholdEvent` monitoring table exceed this value, SAP ASE overwrites the oldest unread events with new events.

threshold event monitoring

Table 8. Summary Information

Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Monitoring

Enable or disables SAP ASE from recording threshold events.

user log cache queue size

Table 9. Summary Information

Default value	1 (on)
Range of values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive

Required role	System Administrator
Configuration group	User Environment

Determines whether a queueing strategy is used for logging. Setting **user log cache queue size** to:

- 1 – enables queuing for user log caches. The user log cache is divided into multiple cachelets, the number of which is dependent on the value of **user log cache size**.
- 0 – disables queuing for user log caches. The user log cache is not divided into multiple cachelets regardless of the value to which you set **user log cache size**.

Changed Configuration Parameters

SAP ASE version 16.0 includes changes for configuration parameters.

allow nested triggers
The multiple triggers feature does not change the behavior of the **sp_configure "allow nested triggers"** configuration parameter.

stack guard size
(On UNIX platforms only) SAP recommends that you include an additional 4096 bytes when you configure stack guard size to a nondefault value to increase the usable portion of the stack gruard area.

Built-In Functions

SAP ASE version 16.0 introduces a new built-in function.

dbencryption_status

SAP ASE 16.0 introduces a new built-in function, **dbencryption_status()**, which supports the full database encryption feature by reporting on the encryption/decryption status and progress of a database.

Reports database encryption/decryption status and progress. The syntax is:

```
dbencryption_status ('status'|'progess', dbid[,
    lstart])
```

where:

- `status` – returns the encryption status of the database you specify in *dbid*. You must supply *dbid* to use `status`. The returned values are:
  - 0 – indicates a normal database.
  - 1 – indicates that a database is encrypted.

  - 2 – indicates that a database is being encrypted.
  - 3 – indicates that a database is partially encrypted (but not in the process of being encrypted).
  - 4 – indicates that a database is being decrypted.
  - 5 – indicates that a database is partially decrypted (but not in the process of being decrypted).
- `progress` – reports on the percentage of encryption/decryption progress. If you supply:
  - *dbid* – `progress` returns the percentage of processed pages in the whole database.
  - Both *dbid* and *lstart* (the logical start page) – `progress` returns the percentage of processed pages in the fragment indicated by *lstart*.

  When you use "`progress`" and SAP ASE finds no progress information, such as when there is no encryption or decryption operation occurring, or if the encryption/decryption process is finished, SAP ASE returns "`-1`."
- *dbid* – is the database ID.

See also

- *Create an Encrypted Database* on page 68
- *create archive database for Full Database Encryption* on page 119
- *create database for Full Database Encryption* on page 119
- *Encrypt an Existing Database* on page 69
- *sp_helpdb* on page 163
- *Suspend the Encryption Process* on page 75
- *Resume the Encryption Process* on page 76
- *alter database for Full Database Encryption* on page 113
- *Full Database Encryption and System Changes* on page 81
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *create encryption key* on page 122
- *drop encryption key* on page 152

Commands

SAP ASE version 16.0 introduces changes to commands.

alter database for Full Database Encryption

In SAP ASE version 16.0, you can use the **alter database** command to encrypt and decrypt databases.

Syntax

```
alter database database_name
{[encrypt with key_name | decrypt [with key_name]] [parallel
degree_of_parallelism]
| resume [encryption | decryption [parallel degree_of_parallelism]]
| suspend [encryption | decryption]
}
```

Parameters

- ***database_name* –** is the name of the database you want to encrypt or decrypt.
- **encrypt with *key_name* –** instructs SAP ASE to fully encrypt the database.

  Specifically, the command retrieves the corresponding key ID from the `sysencryptkeys` system table in the `master` database and set the `encrkeyid` column in its related `sysdatabases` row.

  key_name is the database encryption key you used to encrypt the database. If you do specify a different key name, the command fails and SAP ASE displays an error message.

  SAP ASE fails to run **alter database** and displays an error message if the database is already:

  - Encrypted with another key.
  - Being encrypted.

  If you run this command on a partially encrypted database that is not currently being encrypted, SAP ASE treats the command as if you specified the `resume encryption` option, as long as the key name is the same as the previously specified key.
- **decrypt [with *key_name*] –** instructs SAP ASE to decrypt the database. When decrypting a database, `[with key_name]` is optional, as SAP ASE looks up the key ID in the `sysdatabases` system table. The command fails, however, if you specify *key_name* with a different key name than what was used to encrypt the database.
- **resume decryption –** instructs SAP ASE to resume the decryption process for the database in which an earlier decryption process was suspended. SAP ASE ignores this command if the *database_name* is already completely decrypted.
- **parallel *degree_of_parallelism* –** determines how many worker threads to initiate for the task.

  Create a thread for each database storage virtual device, as long as the number is equal to or fewer than "`number of worker processes`" configuration. The *degree_of_parallelism* number should be no larger than the number of database devices

because additional worker threads do not improve encryption performance. If you do not specify *degree_of_parallelism*, SAP ASE internally defines the value based on the number of online engines, as well as how the database is distributed across various devices.

- **resume encryption –** resumes the encryption process from the page where encryption was previously suspended.

  The command fails if:

  - There is an encryption process already running in SAP ASE.
  - Encryption was never started on the database.
  - The encryption process already completed.

  You can use `parallel` *`degree_of_parallelism`* with `resume encrypt`. If you do not specify `parallel` *`degree_of_parallelism`*, SAP ASE determines the value based on how the database is distributed across various engines.
- **suspend encryption –** terminates all encryption worker threads that are encrypting data. SAP ASE records the progress of encryption so that `resume encryption` can restart encryption where the previous encryption process stopped. SAP ASE ignores this command if there is no encryption in progress.

Examples

- **Example 1 –** This example alters an existing database called `existdb` for encryption using an encryption key called `dbkey`:

  ```
  alter database existdb encrypt with dbkey
  ```

  The example does not specify the parallel degree, leaving it up to SAP ASE to determine how many worker threads should be initiated to encrypt `existdb` in parallel.
- **Example 2 –** This example suspends an encryption operation on a database called `existdb`:

  ```
  alter database existdb suspend encryption
  ```
- **Example 3 –** This example resumes a suspended encryption on a database called `existdb`:

  ```
  alter database existdb resume encryption
  ```

Usage

- Database encryption occurs while the database is running. This means the database is accessible by other users while it is being encrypted; you need not put it into single-user mode.
- The encryption process does not interrupt user queries, updates, or insert operations on the database.
- You can suspend and resume database encryption, so that you can resume encrypting the database after restarting SAP ASE.

- The encryption operation is not transactional.
- You can alter both archive and temporary databases for encryption and decryption.
- SAP ASE records the encryption progress of a database and provides utilities to report its status.

The command fails if:

- You use it on a database that is already encrypted.
- You use it on a database that is being encrypted. If you use this command on a partially encrypted database, but there is no encryption process running in SAP ASE, the command resumes encryption from the location where it was last suspended as long as you use the same database encryption key name in your previous command to encrypt the database.

Restrictions:

- You cannot encrypt the `master` and `model` databases.
- You cannot decrypt a database that is being encrypted, or encrypt a database that is being decrypted.
- You cannot drop or unmount a database when it is being encrypted.
- You cannot load another database on top of a database that is being encrypted.
- You cannot back up (dump) a database while a database is being encrypted.

See also

- *Encrypt an Existing Database* on page 69
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *Suspend the Encryption Process* on page 75
- *Resume the Encryption Process* on page 76
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152
- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177

alter index

Change the compression state of future index inserts or updates using the **index_compression** clause.

Syntax

```
alter index [[database.][owner].table_name.index_name
set index_compression [= {none | page} ]
```

Parameter Changes

index_compression – changing the local index partition's compression state affects only index rows that are newly inserted or updated in the partition.

- `none` – the index page for the specified index is not compressed. Indexes that are specifically created with **index_compression = page** are compressed.
- `page` – when the page is full, existing index rows are compressed using the page prefix compression. When a row is added, a check is performed to determine whether the row is suitable for compression.

Example

sets the compression state to on for the index `idx_char`:

```
alter index order_line. idx_char
    set index_compression = page
```

See also

- *Changing the Compression State* on page 60
- *alter table for Index Compression* on page 116

alter table for Index Compression

Change the compression state of future index inserts or updates using the **index_compression** clause.

Syntax

```
alter table [[database.][owner].table_name
{add column_name datatype}
   [default {constant_expression | user | null}]
    {identity | null | not null}
    [off row | in row]
    [[constraint constraint_name]
       {{unique | primary key}
          [clustered | nonclustered] [asc | desc]
        [with {fillfactor = pct,
               max_rows_per_page = num_rows,
               reservepagegap = num_pages}]
        [on segment_name]
| references [[database.]owner.]ref_table
```

```
      [(ref_column)] [match full]
| check (search_condition)]
[encrypt [with key_name] [decrypt_default value]],
[[not] compressed]
[, next_column]...
| add {[constraint constraint_name]
        {unique | primary key}
           [clustered | nonclustered]
           (column_name [asc | desc]
        [, column_name [asc | desc]...])
           [with {fillfactor = pct,
                  max_rows_per_page = num_rows,
                  reservepagegap = num_pages}]
           [on segment_name]
| foreign key (column_name [{, column_name}...])
     references [[database.]owner.]ref_table
        [(ref_column [{, ref_column}...])]
   [match full]
| check (search_condition)}
| set {  [ dml_logging = {full | minimal | default}]
       | [,compression = {NONE | PAGE | ROW | ALL} ]
       | [,index_compression = {NONE | PAGE} ]
      }
| drop {column_name [, column_name]...
| constraint constraint_name}
| modify column_name
     [datatype [null | not null]]
     [[[encrypt [with key_name]
         [decrypt_default value]]
        | decrypt
     ]
     [[not] compressed ]
[, next_column]...
| replace column_name
      default {constant_expression | user | null}
| decrypt_default {constant_expression | null}
| drop decrypt_default
| lock {allpages | datarows | datapages}
| with exp_row_size=num_bytes
| partition number_of_partitions
| unpartition
| partition_clause
| alter_partition_clause


     alter_partition_clause ::=
{add_partition_clause
| drop_partition_clause
| modify partition partition_name
            [, partition_name ...]
   set compression [= {none | row | page | ALL} ]
   set index_compression [= {none | page} ]
```

Parameter Changes

index_compression – specifies the index compression to be enabled or disabled to the table, index, or the local index partition. If the table is modified to be index compressed, newly created indexes are compressed.

- `NONE` – indexes on the specified table are not compressed.
- `PAGE` – all indexes on the specified table are compressed.

Examples

Example 1

Alters the existing table `order_line`, changing the compression state to `NONE`:

```
alter table order_line set index_compress = NONE
```

Example 2

Alters the existing table `sales`, changing the state to `PAGE` for compression of the local index partition Y2009.

```
alter table sales
modify partition Y2009 set index_compression = PAGE
```

See also

- *Changing the Compression State* on page 60
- *alter index* on page 116

alter table for Multiple Triggers

If there are multiple triggers on a table, the table owner can disable any or all of the multiple triggers defined on that table.

alter table for Residual Data Removal

The **alter table** command supports the ability to remove residual data from deletions in SAP ASE.

Syntax

The syntax to specify this in a table that already exists is:

```
alter table table_name
    set "erase residual data" {on | off}
```

Usage

When you set this option on a table, the operations for the table (**drop table**, **delete row**, **alter table**, **drop index**) that result in residual data automatically clean up deallocated space. The default is set to **off**.

Permissions

Only the table owner or a user with **alter any table** permission can use the "`erase residual data`" option.

create archive database for Full Database Encryption

The **create archive database** command supports the full database encryption feature.

Syntax

```
create archive database database_name
    encrypt with key_name
```

Parameters

- *database_name* is the name of the archive database you are creating.
- *key_name* is the same key that you used to encrypt the database that was backed up (dumped). SAP ASE verifies that *key_name* matches during the database dump. If it does not match, the restoration fails.

Permissions

There is no special permission to use the `encrypt with` option of the **create archive database** command. However, users need **select** permission on the database encryption key to be able to reference it as the *key_name*.

See also

- *Create an Encrypted Database* on page 68
- *dbencryption_status* on page 111
- *create database for Full Database Encryption* on page 119
- *Full Database Encryption and System Changes* on page 81
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *alter database for Full Database Encryption* on page 113
- *create encryption key* on page 122
- *drop encryption key* on page 152

create database for Full Database Encryption

You can use the **create database** command to create a fully encrypted database.

Specify whether to encrypt a database when you create it, and all the data inserted into the database becomes encrypted automatically. The size of the database does not change when it is

encrypted, and all storage access functions work identically whether a database is encrypted or not. The types of databases that support encryption are:

- Normal user database
- Temporary database
- Archive database

You cannot encrypt an in-memory database.

Syntax

```
create [temporary] database database_name
    encrypt with key_name
```

Parameters

- ***database_name* –** is the name of the encrypted database you are creating.
- ***key_name* –** is the name of the database encryption key.

Examples

- **Create Encrypted Database from Scratch –** Creates an encrypted database called `demodb` with a log called `demodev` on a machine called `demologdev`, using an encrypttion key called `dbkey`:

  ```
  create database demodb on demodev log on demologdev encrypt with
  dbkey
  ```

See also

- *Create an Encrypted Database* on page 68
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *alter database for Full Database Encryption* on page 113
- *create encryption key* on page 122
- *drop encryption key* on page 152
- *Full Database Encryption and System Changes* on page 81
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177

create default

The **or replace** clause allows you to replace a default's definition using **create default**.

Syntax

Changes are in bold.

```
create [or replace] [owner.] default_name
as constant_expression
```

*Parameter Changes for **create or replace default***

- **create** – creates a default if one does not already exist.
- **or replace** – if the default already exists, replaces the default definition with the new definition.
- *default_name* – if the specified default name already exists, then it is replaced with the new default definition. The object name and ID remain the same.
- *constant_expression* – the definition of the default can be changed when the default is replaced. The new default value overrides the old default.

Example

This example creates a default with a phone number defined as `UNKNOWN`:

```
create default phonedflt as "UNKNOWN"

select object_id("phonedflt")
-----------
1001051571
```

This default replaces the previously created default using the **or replace** clause. The phone number is changed, but the object ID of the default remains the same:

```
create or replace default phonedflt as "999-999-9999"

select object_id("phonedflt")
-----------
1001051571
```

Objects Dependent on Replaced Defaults

- Many columns can be bound to a replaced default.
- User defined datatypes can be bound to the replaced defaults.

Procedures that access these columns will be recompiled when the default is replaced and the procedure is executed.

*Permission Changes for **create or replace default***

Any user who impersonates the default owner through an alias or **setuser** cannot replace the default.

Changes for replacing a default are in bold.

Granular permissions enabled	With granular permissions enabled, you must have the `create default` privilege. To create a default for another user, you must have the `create any default` privilege. **You must be the default owner to replace the default.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner, a user with **sa_role**, or have the `create default` privilege. **You must be the default owner to replace the default.**

*Auditing Changes for **create or replace default***
Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
14	create	create default	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create encryption key

The **create encryption key** command supports the full database encryption feature.

The database encryption key is a 256-bit symmetric key that is created in the master database and used to encrypt a database.

Syntax

```
create encryption key keyname
    [for algorithm]
    for database encryption
    [with
```

```
        {[master key]
        [key_length 256]
        [init_vector random]
        [[no] dual_control]}
```

Parameters

- ***keyname*** – Must be unique in the user's table, view, and procedure namespace in the current database. Specify the database name if the key is in another database, and specify the owner's name if more than one key of that name exists in the database. The default value for owner is the current user, and the default value is the current database. Only the system security officer can create keys for other users.
- ***algorithm*** – Specifies the algorithm. The only algorithm supported is Advanced Encryption Standard (AES). AES supports key sizes of 128, 192, and 256 bits, and a block size of 16 bytes. The block size is the number of bytes in an encryption unit. Large data is subdivided for encryption.
- **for database encryption** – Indicates that you are creating an encryption key to encrypt an entire database, rather than a column.
- **master key** – Creates a master key in the `master` database, and indicates to SAP ASE to protect the database encryption key using that key. By default, SAP ASE uses this master key (if it exists) to protect database encryption keys.
- **key_length 256** – Is the size, in bits, of the key you are creating. The only valid length for a database encryption key is 256; you see an error message if you use any other size.
- **init_vector random** – Specifies the use of an initialization vector during encryption. When the encryption algorithm uses an initialization vector, the cipher text of two identical pieces of plain text are different, which prevents detection of data patterns. Using an initialization vector can add to the security of your data. Database encryption enforces stronger security than column encryption; if you specify `init_vector null` as you can for creating a column encryption key, SAP ASE returns an error.
- **[no] dual control** – Indicates whether the new key must be encrypted using dual controls. By default, dual control is not configured. Both the master key and dual master key must exist in the `master` database to use `dual control`.

Examples

- **Example 1 –** Creates a master key to protect "testkey":

  ```
  create encryption key testkey for database encryption
      with master key
  ```

- **Example 2 –** Creates a dual master key to protect "testkey":

  ```
  create encryption key testkey for database encryption
      with dual_control
  ```

- **Example 3 –** Creates both a master key and dual master key to protect "testkey":

```
create encryption key testkey for database encryption
    with master key dual_control
```

- **Example 4 –** Creates a master key to protect "testkey", while explicitly excluding the dual master key:

```
create encryption key testkey for database encryption
    with master key no dual_control
```

- **Example 5 –** Creates a master key to protect "testkey" while explicitly excluding a dual master key:

```
create encryption key testkey for database encryption
    with no dual control
```

- **Example 6 –**

```
sp_configure 'enable encrypted columns', 1
create encryption key master with passwd "testpassword"
set encryption passwd 'testpassword' for key master
create encryption key dbkey for database encryption
```

Usage

- The database encryption key does not support the `pad` option in **create encryption key** command.
- The database encryption key cannot be the default key for column encryption.
- Successfully created database encryption keys are stored in the `sysencryptkeys` table of the `master` database and are indicated by this key type:

```
#define EK_DBENCKEY       0x1000
```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

The permission checks for **create encryption key** differ, based on your granular permission settings:

Granular permissions enabled	SAP ASE creates a new permission called "`manage database encryption key`." You must have permission to create a database encryption key.
Granular permissions disabled	You must be a user with sso_role, keycustodian_role, or have **create encryption key** privilege.

See also

- *drop encryption key* on page 152
- *Changing a Database Encryption Key* on page 67
- *Dropping a Database Encryption Key* on page 68

- *Back Up the Database Encryption Key* on page 78
- *Creating the Database Encryption Key* on page 65
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177

create function

The **or replace** clause allows you to replace a user-defined function's definition using **create function**.

Syntax

Changes are in bold.

```
create [or replace] function
    [ owner_name.] function_name
    [ ( @parameter_name [as] parameter_datatype
    [ = default ][ ,...n ] ) ]
    returns  return_datatype
    [ with recompile]
    as
    [begin]
    function_body
    return scalar_expression
    [end]
```

*Parameter Changes for **create or replace function***

- **create** – creates a function if one does not already exist.
- **or replace** – re-defines an existing function. Use this clause to change the definition of an existing user defined function without dropping, re-creating, and regranting object privileges previously granted on the function. If the function is redefined, it is recompiled when the function is used.
- *function_name* – the name of the function remains the same, although its definition is changed.
- **@parameter_name** – names and number of parameters can be altered when the function definition is replaced.
- *parameter_datatype* – you can change the datatype of the parameter to the function.

- **with recompile** – you can change the option to recompile, or not to recompile every time the function is replaced.
- *return_datatype* – you can alter the return datatype of the function.
- *scalar_expression* – you can change the value returned by the function.
- *function_body* – you can change the SQL statements that define the value of the function.

Example

This example defines a function which concatenates `firstname` and `lastname` strings.

```
create function fullname(
    @firstname char(30),
    @lastname char(30))
returns char(61)
as
begin
declare @name char(61)
set @name = @firstname|| ` ` ||@lastname
return @name
end

select object_id("fullname")
-----------
473049690
```

This function replaces the previously created `fullname` function using the **or replace** clause. After replacing the function, the local variable `@name` is removed. The object ID of the function remains the same.

```
create or replace function fullname(
    @firstname char(30),
    @lastname char(30))
returns char(61)
as
begin
return(@firstname|| ' ' ||@lastname)
end

select object_id("fullname")
-----------
473049690
```

Objects Dependent on Replaced Functions

If the replaced function is called by another function, both functions will be recompiled when called. If the interface of the replaced function does not match that in the calling function, then the calling function must be replaced, otherwise the calling function raises an error. You can execute **sp_depends** on the replaced function to check for any calling objects.

For example, `testfun1` is replaced to have two parameters instead of one. The calling function, `testfun2`, must be replaced to account for the second parameter.

```
create function testfun1 (@para1 int)
    returns int
```

```
    as
    begin
        declare @retval int
        set @retval = @para1
        return @retval
    end
create function testfun2 (@para int)
    returns int
    as
    begin
        declare @retval int
        select @retval= dbo.testfun1 (@para)
        return @retval
    end
create or replace function testfun1 (@para1 int,@para2 int)
    returns int
    as
    begin
        declare @retval int
        set @retval = @para1+@para2
        return @retval
    end
```

Restrictions

If a function is referenced in a computed column or functional index, it cannot be replaced.

*Permission Changes for **create or replace function***

Any user who impersonates the function owner through an alias or **setuser** cannot replace the function.

Changes for replacing a function are in bold.

Granular permissions enabled	With granular permissions enabled, you must have the `create function` privilege. You must have the `create any function` privilege to run **create function** for other users. **You must be the function owner to replace the function.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner or have the `create function` privilege. **You must be the function owner to replace the function**.

*Auditing Changes for **create or replace function***

Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
97	create	create function	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create function (SQLJ)

The **or replace** clause allows you to replace a user-defined SQLJ function's definitions using **create function**.

Syntax

Changes are in bold.

```
create [or replace] function
    [owner_name.]sql_function_name
        ([ sql_parameter_name sql_datatype
            [(length)| (precision[, scale ])]
        [[,sql_parameter_name sql_datatype
            [(length)| (precision[, scale])]]
        ...]])
    returns sql_datatype
    [(length)| (precision[, scale])]
    [modifies sql data]
    [returns null on null input |    called on null input]
    [deterministic | not deterministic]
    [exportable]
    language java
    parameter style java
    external name 'java_method_name
        [([java_datatype[, java_datatype
        ...])] '
```

Parameter Changes for create or replace function (SQLJ)

- **create** – creates a SQLJ function if one does not already exist.

- **or replace** – re-defines an existing function. Use this clause to change the definition of an existing user defined SQLJ function without dropping, re-creating, and regranting object privileges previously granted on the function.
- *sql_function_name* – the name of the function remains the same, although its definition is changed.
- **sql_parameter_name** – names and the number of parameters can be altered when the function definition is replaced.
- *sql_datatype* – the Transact-SQL datatype of the parameter to the function can be changed.
- **returns** *sql_datatype* – result datatype of the function can be changed.
- **external** – name of external routine can be changed.
- *java_method_name* – the Java method name can be changed.
- *java_datatype* – the Java datatype can be changed.

Example

This example creates a SQLJ function named `sqlj_testfun`.

```
create function sqlj_testfun (p1 int)
                      returns int
                      language java
                      parameter style java
                      external name 'UDFSample.sample(int)'
```

The following replaces the previously created SQLJ function using the **or replace** clause. Parameter `p2` is added and the external java method is changed but the object ID of the SQLJ function remains the same.

```
create or replace function sqlj_testfun (p1 int,p2 int)
                      returns int
                      language java
                      parameter style java
                      external name 'UDFSample.sample2(int,int)'
```

Objects Dependent on Replaced Functions

If the replaced SQLJ function is called by another function, both functions will be recompiled when called.

If the interface of the replaced function does not match that in the calling function, then the calling function must be replaced, otherwise the calling function raises an error. You can execute **sp_depends** on the replaced function to check for any calling objects.

Restrictions

If a function is referenced in a computed column or functional index, it cannot be replaced.

*Permission Changes for **create or replace function** (SQLJ)*

Any user who impersonates the function owner through an alias or **setuser** cannot replace the function.

Changes for replacing a function are in bold.

Granular permissions enabled	With granular permissions enabled, you must have the `create function` privilege. You must have the `create any function` privilege to run **create function** for other users. **You must be the function owner to replace the function.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner or have the `create function` privilege. **You must be the function owner to replace the function**.

*Auditing Changes for **create or replace function** (SQLJ)*
Changes are in bold.

Event	Audit Option	Command or access audited	Information in **extrainfo**
97	create	create function	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create index

An index or index partition can be compressed with the **index_compression** clause.

Syntax

```
create [unique] [clustered | nonclustered] index index_name
    on [[database.]owner.]table_name
    (column_expression [asc | desc]
    [, column_expression [asc | desc]]...)
    [with {fillfactor = pct,
    index_compression = { NONE | PAGE },
    max_rows_per_page = num_rows,
```

```
    reservepagegap = num_pages,
    consumers = x, ignore_dup_key, sorted_data,
    [ignore_dup_row | allow_dup_row],
    statistics using num_steps values}]
    [on segment_name]
[index_partition_clause]
Syntax to create index partitions
index_partition_clause::=
[local index
[partition_name [on segment_name]
[with index_compression = { NONE | PAGE }]
[, partition_name [on segment_name]
[with index_compression = { NONE | PAGE }]...]]]
```

Parameter Changes

index_compression

- NONE – the index page for the specified index is not compressed. Indexes that are specifically created with **index_compression = PAGE** are compressed.
- PAGE – when the page is full, existing index rows are compressed using the page prefix compression. When a row is added, a check determines whether the row suitable for compression.

Examples

Example 1

Creates a compressed index called idx_order_line on columns ol_delivery_d and ol_dist_info:

```
create index idx_order_line
    on order_line (ol_delivery_d, ol_dist_info)
with index_compression = page
```

If the index has an index row length that is too short to benefit from compression, a warning is raised indicating the index will not be compressed.

Example 2

Creates a compressed index called idx_Sales. The index contains local index partitions that can be compressed. Index prefix compression is applied to the local index partition. Page prefix compression is applied while the index page is full:

```
create index idx_sales
    on Sales(store_id, order_num)
    local index ip1 with index_compression = PAGE,
    ip2 with index_compression = PAGE,
ip3
```

See also

- *Creating a Compressed Index* on page 59

create procedure

The **or replace** clause allows you to replace a procedure's definition using **create procedure**.

Previously granted privileges on a replaced procedure are preserved.

Syntax

Changes are in bold.

```
create [or replace] procedure

    [owner.]procedure_name[;number]
    [[(@parameter_name datatype [(length)
    (precision[,scale])]
    [= default][output]…)]]
[with {recompile | execute as {owner | caller}} ]
as {SQL_statements | external name dll_name}
```

*Parameter Changes for **create or replace procedure***

- **create** – if only **create** is specified, a new procedure is created.
- **or replace** – if the specified procedure does not exist, a new procedure is created. If the procedure does exist, the procedure definition is changed; existing permissions, auditing options, and replication attributes are preserved.
- *procedure_name* – if the procedure is being replaced, the name of the procedure remains the same and the object identifier of the procedure in all the respective catalogs remains the same.
- *number* – if **or replace** is not specified and the procedure exists, SAP ASE raises an error that the procedure has already been created with that group number, and you must create the procedure with a different group number. If **or replace** clause is specified, but the group number is not, but the procedure exists with different group numbers, an error is raised because the procedure is part of a group and it cannot be replaced. You cannot specify the group number to replace an existing procedure with that group number. This is similar to the dropping of procedures where individual procedures within a group cannot be dropped.
- *parameter_name* – you can change the name and number of parameters when the procedure definition is replaced.
- *datatype* [(*length*) (*precision*[,*scale*])] – you can change the type, length, precision and scale of the parameters.
- *default* – you can change the default to NULL for the parameters, or set a different value when the procedure is replaced.
- **output** – you can change the return parameter of the procedure.
- **with recompile** – if an existing procedure has been created with this option, then it can be changed using the **or replace** clause so that SAP ASE does not create a new plan each time the procedure is executed. If the existing procedure has not been created using **with**

recompile, then it can be replaced with the new definition so that the plan is created each time the procedure is executed.

- **with execute as** – an existing procedure's **with execute as** clause can be changed from owner to caller and vice versa. You can also re-create a procedure without the **with execute as** clause using the **or replace** clause.
- *SQL_statements* – you can change the body of the procedure to contain statements different from those in the existing procedure.
- **external name** – extended stored procedures can also be replaced.
- *dll_name* – the name of the dynamic linked library that implements the extended stored procedures can be changed.

Example

This example is based on a table of information about products, defined as:

```
create table Products (
    ProductID int,
    ProductName varchar(30),
    Discontinued varchar(10))

create procedure ProductType
    @product_ID int,
    @type char(10) output
as
declare @prod_name char(20)
select @prod_name = ProductName, @type =
    case @prod_name
        when 'Tee Shirt' then 'Shirt'
        when 'Sweatshirt' then 'Shirt'
        when 'Baseball Cap' then 'Hat'
        when 'Visor' then 'Hat'
        when 'Shorts' then 'Shorts'
        else 'UNKNOWN'
    end
from Products
where ProductID = @product_ID

select object_id("ProductType")
-----------
425049519
```

This next command replaces the `ProductType` procedure using the **or replace** clause. The parameters for `Tee Shirt` and `Sweatshirt` are updated, but the object ID of the procedure remains the same.

```
create or replace procedure ProductType
    @product_ID int,
    @type char(10) output
as
declare @prod_name char(20)
select @prod_name = ProductName, @type =
    case @prod_name
        when 'Tee Shirt' then ' T Shirt'
```

```
        when 'Sweatshirt' then 'Long Sleeve Shirt'
        when 'Baseball Cap' then 'Hat'
        when 'Visor' then 'Hat'
        when 'Shorts' then 'Shorts'
        else 'UNKNOWN'
    end
from Products
where ProductID = @product_ID

select object_id("ProductType")
-----------
425049519
```

Objects Dependent on Replaced Procedures

A procedure that calls a replaced procedure is recompiled when it executes. If replacing the procedure changed the number or type of parameters, the calling procedure must be replaced. You can run **sp_depends** on the replaced procedure to verify whether there are calling procedures that are affected by the changed definition.

*Permission Changes for **create or replace procedure***

Any user who impersonates the procedure owner through an alias or **setuser** cannot replace the procedure.

Changes for replacing a procedure are in bold.

Granular permissions enabled	When granular permissions is enabled, you must have the `create procedure` privilege. You must have the `create any procedure` privilege to run **create procedure** for other users. **You must be the procedure owner to replace the procedure.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner or have the `create procedure` privilege. **You must be the procedure owner to replace the procedure.**

*Auditing Changes for **create or replace procedure***

Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
11	create	create procedure	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create procedure (SQLJ)

The **or replace** clause allows you to replace a SQLJ procedure definitions using **create procedure**.

Previously granted privileges on a replaced procedure are preserved.

Syntax

Changes are in bold.

```
create [or replace] procedure

    [owner_name.]sql_procedure_name
        ([[in | out | inout]  sql_parameter_name
        sql_datatype
            [(length)| (precision[, scale ])]
            [=default]
        ...])
        [,[in | out | inout]  sql_parameter_name
        sql_datatype
            [(length)| (precision[, scale])]]
            [=default]
        ...])
    [modifies sql data]
    [dynamic result sets integer]
    [deterministic | not deterministic]
    language java
    parameter style java
    external name 'java_method_name
        [([java_datatype[, java_datatype
        ...]])] '
```

*Parameter Changes for **create or replace procedure** (SQlJ)*

- **create** – if only **create** is specified, a new SQLJ procedure is created.
- **or replace** – if the specified SQLJ procedure does not exist, a new SQLJ procedure is created. If the SQLJ procedure does exist, the definition is changed.
- *sql_procedure_name* – name and number of parameters can be changed.
- **in** | **out** | **inout** – the mode of the listed parameter can be changed.
- *sql_datatype* [(*length*) (*precision*[,*scale*])] – you can change the type, length, precision and scale of the parameters.
- *default* – you can change the default to NULL for the parameters, or set a different value when the procedure is changed.
- **deterministic** | **not deterministic** – you can change the deterministic value.
- **external** – name of external routine can be changed.
- *java_method_name* – the Java method name can be changed.
- *java_datatype* – the Java datatype can be changed.

Example

This example creates a SQLJ procedure named `proc_name`.

```
create procedure sqlj_proc (param int)
                              language java
                              parameter style java
                         external name 'UDFSample.sample(int)'
```

This procedure replaces the previously created SQLJ procedure using the **or replace** clause. Parameter `p2` is added and the external java method is changed but the object ID of the SQLJ procedure remains the same.

```
create or replace procedure sqlj_proc (p1 int, p2 int)
                              language java
                              parameter style java
                         external name 'UDFSample.add(int,int)'
```

Objects Dependent on Replaced Procedures

A Transact-SQL procedure that calls a replaced SQLJ procedure is recompiled when it executes. If replacing the SQLJ procedure changed the number or type of parameters, the calling procedure must be replaced. You can run **sp_depends** on the replaced procedure to verify whether there are calling procedures that are affected by the changed definition.

*Permission Changes for **create or replace procedure** (SQLJ)*

Any user who impersonates the procedure owner through an alias or **setuser** cannot replace the procedure.

Changes for replacing a SQLJ procedure are in bold.

Granular permissions enabled	When granular permissions is enabled, you must have the `create procedure` privilege. You must have the `create any procedure` privilege to run **create procedure** for other users. **You must be the procedure owner to replace the procedure.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner or have the `create procedure` privilege. **You must be the procedure owner to replace the procedure.**

*Auditing Changes for **create or replace procedure** (SQLJ)*
Changes are in bold.

Event	Audit Option	Command or access audited	Information in **extrainfo**
11	create	create procedure	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create rule

The **or replace** clause allows you to replace a rule's definition using **create rule**.

Syntax
Changes are in bold.

```
create [or replace] [[and | or] access] rule
    [owner.]rule_name
    as condition_expression
```

*Parameter Changes for **create or replace rule***

- **create** – creates a rule if one does not already exist.
- **or replace** – if the rule already exists, replaces the rule definition with the new definition.
- **access** – an access rule can be changed from an “and” rule to an “or” rule, and vice versa. Access rules cannot be replaced with a domain rule of the same name, and vice versa.
- *rule_name* – if the specified rule name already exists, it is replaced with the new rule definition, but the name is preserved.
- *constant_expression* – you can change the definition of the rule when the rule is replaced. The new rule value overrides the old rule value.

Examples

Example 1

This example creates a rule named `limit`, which limits the value of advance to $1000:

```
 create rule limit
    as @advance < $1000

select object_id("limit")
 -----------
1017051628
```

This next command replaces the created rule. The limit is changed using the **or replace** clause. The object ID of the rule remains the same.

```
 create or replace rule limit
    as @advance < $2000

select object_id("limit")
 -----------
1017051628
```

Example 2

The table owner creates an **AND access rule** called `uname_acc_rule`:

```
create access rule uname_acc_rule
as @username = suser_name()

select object_id("uname_acc_rule")

-----------
1033051685
```

Replace `uname_acc_rule` with an **OR access rule**:

```
create or replace or access rule uname_acc_rule
as @username = suser_name()

select object_id("uname_acc_rule")
```

```
-----------
1033051685
```

Objects Dependent on Replaced Rules

- Columns from many tables can be bound to the replaced rules.
- User defined datatypes can be bound to the replaced rules.

Procedures that access these columns will be recompiled when the rule is replaced and the procedure is executed.

*Permission Changes for **create or replace rule***
Any user who impersonates the rule owner through an alias or **setuser** cannot replace the rule.

Changes for replacing a rule are in bold.

Granular permissions enabled	With granular permissions enabled, you must have the `create rule` privilege. You must have the `create any rule` privilege to use **create rule** for other users. **You must be the rule owner to replace the rule.**
Granular permissions disabled	With granular permissions disabled, you must have the `create rule` privilege, be the database owner, or a user with sa_role. You must be a user with sa_role to use **create rule** for other users. **You must be the rule owner to replace the rule**.

*Auditing Changes for **create or replace rule***
Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
13	create	create rule	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create table for Index Compression

Indexes on a specified table can be compressed with the **index_compression** clause.

Syntax

```
create table [database.[owner].]table_name
(column_name datatype
[default {constant_expression | user | null}]
{[{identity | null | not null}]
[off row | [in row [(size_in_bytes)]]
[[constraint constraint_name]
{{unique | primary key}
[clustered | nonclustered] [asc | desc]
[with {fillfactor = pct,
max_rows_per_page = num_rows,}
reservepagegap = num_pages}]
[on segment_name]
| references [[database.]owner.]ref_table
[(ref_column )]
[match full]
| check (search_condition)}]}
[match full]...
[encrypt [with key_name]
    [decrypt_default constant_expression | null]]
[[constraint [[database.[owner].]key_name]
    {unique | primary key}
        [clustered | nonclustered]
        (column_name [asc | desc]
        [{, column_name [asc | desc]}...])
        [with {fillfactor = pct
```

```
             max_rows_per_page = num_rows,
            reservepagegap = num_pages}]
    [on segment_name]
| foreign key (column_name [{,column_name}...])
references [[database.]owner.]ref_table
[(ref_column [{, ref_column}...])]
[match full]
| check (search_condition) ...}
 [{, {next_column | next_constraint}}...]
 [lock {datarows | datapages | allpages}]
[with {max_rows_per_page = num_rows,
    exp_row_size = num_bytes,
    reservepagegap = num_pages,
    identity_gap = value,
    transfer table [on | off],
    compression [={NONE | ROW | PAGE}],
    index_compression [={NONE | PAGE} ]
      }
]
[on segment_name]
     [[ external table ] at pathname ]
[partition_clause]
```

Parameter Changes

index_compression

- NONE – indexes on the specified table are not compressed. Indexes that are specifically created with **index_compression = PAGE** are compressed.
- PAGE – all indexes on the specified table are compressed. Indexes that are specifically created with **index_compression = NONE** are not compressed.

Example

This example creates the index compressed table `order_line` with columns `ol_delivery_d` and `ol_dist_info` compressed and using page-level compression:

```
create table order_line (
    ol_o_id         int,
    ol_d_id         tinyint,
    ol_w_id         smallint,
    ol_number       tinyint,
    ol_i_id         int,
    ol_supply_w_id  smallint,
    ol_delivery_d   datetime,
    ol_quantity     smallint,
    ol_amount       float,
    ol_dist_info    char(24) )
lock datapages
    with index_compression = page
```

By default, indexes created on this table are compressed by default. However, if an index has an index row length that is too short to benefit from compression, a warning is raised, indicating that the index will not be compressed.

See also

- *Creating an Index Compressed Table* on page 58
- *select into* on page 155

create table for Residual Data Removal

The **create table** command supports the ability to remove residual data from deletions in SAP ASE.

Syntax

The syntax to specify this in a new table is:

```
create table table_name
    with "erase residual data" {on | off}
```

Examples

The following examples use these two tables:

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

Example 1

The option to erase residual data is turned on for table `t1` because it is set at the database level, so that both the **drop table** and **truncate table** commands for `t1` result in the cleanup of all residual data from its pages.

Table `t2`, however, has the **erase residual data** option turned off explicitly, as it was created with the "**erase residual data off**" clause. Residual data is not removed, even though the "`erase residual data`" option is set to `true` at the database level. As a result, residual data remains, even after running **drop table** and **truncate table** on `t2`:

```
create database db1
go
sp_dboption db1, "erase residual data", true
go
use db1
go
create table t1 (col int)
go
insert t1 values ...
go
create table t2 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
drop table t2
go
```

Example 2

In this example:

- Table `t1` does not have "`erase residual data off`" set explicitly, but does have it set at the database level, resulting in the removal of residual data from `t1` when you run **truncate table t1**.
- Table `t2` has the "`erase residual data`" option set at creation, because the option was set at the database level. This results in the removal of residual data from `t2` when you run **truncate table t2**.
- Table `t3` is marked with "`erase residual data off`" explicitly, so that even though **sp_dboption** sets "`erase residual data`" to true, residual data is not removed when SAP ASE runs **truncate table t3**.

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

Example 3

In this example:

- Although both `t1` and `t2` tables had the "`erase residual data` option not set by default, because "`erase_residual_data` was turned on at the session level just before the **truncate table** command was executed, the residual data is removed on both `t1` and `t2`.
- Although table `t3` has the "`erase residual data`" option explicitly set to `off`, residual data is still removed when the **truncate** command is executed, because the "`erase_residual_data`" option is set at the session level.

```
create database db1
go
use db1
go
create table t1(col int)
go
```

```
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
set erase_residual_data on
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

Usage

When you set this option on a table, the operations for the table (**drop table**, **delete row**, **alter table**, **drop index**) that result in residual data automatically clean up deallocated space.

Permissions

Only the table owner or a user with **create any table** permission can use the "**erase residual data**" option.

See also

- *set* on page 156
- *sp_dboption* on page 159

create trigger for Multiple Triggers

Create multiple triggers using the **create trigger** command, and specify the order that the triggers fire with the new `order` parameter.

Syntax

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    {for | instead of} {insert | update | delete}
    [order integer]
    as sql_statements
```

Parameters

`order` *integer* specifies a partial or full ordering of trigger firing:

- Full ordering occurs when you create all the triggers using the `order` clause.
- Partial ordering occurs if you do not specify the `order` clause on some of the triggers. Triggers without the `order` clause implicitly take order number 0 and do not have a defined order, except that they fire after those triggers created using `order`.

Usage

Note: You can only use the `order integer` clause with `for {insert | update | delete}`; you cannot use it with `instead of {insert | update | delete}` triggers.

If you use a duplicate number for `order`, SAP ASE reports an error. `order` numbers need not be consecutive; in fact, nonconsecutive numbers might be preferable, as they allow you to insert new triggers into the middle of an order.

See also

- *Creating Multiple Triggers* on page 89

create trigger for or replace

The **or replace** clause allows you to replace a trigger's definition using **create trigger**.

In versions earlier than SAP ASE 16.0, consecutive **create trigger** commands dropped the old trigger and replaced it with a new trigger definition. However, the auditing options for the trigger were also dropped. Using the optional **or replace** clause, the definition is replaced, and auditing options are preserved.

Syntax

Changes are in bold.

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    {for {insert , update}
        | instead of {insert, update, delete}}
     [order integer]
     [as
        [if update (column_name)
        [{and | or} update (column_name)]...]

            SQL_statements] ...]
        [if update (column_name)
        [{and | or} update (column_name)]...
            SQL_statements]...]
```

*Parameter Changes for **create or replace trigger***

create – creates a trigger if one does not already exist.

- In SAP ASE version 16.0 and later, when there are multiple triggers, specifying **create** without **or replace** raises an error if the trigger name is same. If you specify a different trigger name, a new trigger is created and the old trigger remains. Also see, *Multiple Triggers*

- In versions of SAP ASE earlier than 16.0, if an existing trigger was replaced with the new trigger definition by specifying create without or replace, the name of the new trigger did not need to be same as the name of the old trigger name.
- **or replace** – re-creates an existing trigger. Use this clause to change the definition of a trigger. When specifying **or replace**, auditing options on the trigger are not dropped. If there is no existing trigger with the name you enter, a new one is created and the old trigger remains. This is in conjunction with multiple triggers .
- *trigger_name* – the name of the trigger is not changed when the trigger definition is replaced. The name of the new trigger definition must match the old name to be replaced. If the trigger name differs from any existing trigger, a new trigger is created and the old trigger is not dropped .
- *table_name* – you cannot change the name of the table when a trigger is replaced. If an existing trigger is modified to associate the trigger with another table, then an error is raised indicating that the trigger already exists on another table and cannot be replaced.
- **for | instead of** – you cannot change an "instead of" trigger to a "for" trigger, and vice versa.
- **insert,update,delete** – you can change these actions when you use the **or replace** clause. For example, if the old trigger definition specifies all clauses, the replacement definition can specify all clauses, or a combination of the actions.
- *SQL_statements* – you can change trigger conditions and actions when the trigger definition is replaced.
- **if update** – you can drop or add the **if update** and change the column name referenced by this clause.
- **order** *integer* – the order of the trigger firing can also be changed when the trigger definition is replaced.

Example

This example creates a trigger that prints a message when anyone tries to insert or update data in the `titles` table:

```
create trigger reminder
on titles
for insert, update as
print "Don't forget to print a report for accounting."

select object_id("reminder")
-----------
1312004674
```

The next command changes the message of the printed trigger when anyone tries to update data in the titles table using the **or replace** clause:

```
create or replace trigger reminder
on titles
for update as
print "Don't forget to give a report to accounting."

select object_id("reminder")
```

```
-----------
1312004674
```

*Permission Changes for **create or replace trigger***

Any user who impersonates the trigger owner through an alias or **setuser** cannot replace the trigger.

Changes for replacing a trigger are in bold.

Granular permissions enabled	With granular permissions enabled, you must be the table owner and the `create trigger` privilege must not have been revoked. You must have the `create any trigger` privilege to run **create trigger** on another user's table. **You must be the trigger owner to replace the trigger.**
Granular permissions disabled	With granular permissions disabled: Only a system security officer can grant or revoke permissions to create triggers. The database owner has implicit permission to create a trigger on any user table. Users can create triggers only on tables that they own. The system security officer may revoke user permission to create triggers. Revoking permission to create triggers affects only the database in which the systems security officer issues the **revoke** command. Permission to run the **create trigger** command is restored to the users whose permission was revoked when the system security officer explicitly grants them `create trigger` permission. **You must be the trigger owner to replace the trigger**.

*Auditing Changes for **create or replace trigger***

Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
12	create	create trigger	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

create view

The **or replace** clause allows you to replace a view's definition using **create view**.

Syntax

Changes are in bold.

```
create [or replace] view [owner.]view_name
    [(column_name[, column_name]...)]
    as
    select [distinct] select_statement
    [with check option]
```

*Parameter Changes for **create or replace view***

- **create** – creates a view if one does not already exist.
- **or replace** – replaces an existing view definition without changing any of a view's security attributes.
- *view_name* – is the name of the view being replaced. Only an existing view can be replaced. The object name and ID remain the same.
- *column_name* – specifies names to be used as headings as columns in the view. With the **or replace** clause, you can change column names for the view as follows:
  - If the previous definition of the view contained headings for columns names, then the new definition of the view can omit the headings, or have different headings for column names.

  - If the previous definition of the view did not contain headings for column names, the new definition can contain headings for the view column names.
  - You can change the number of column headings according to the column names in the **select_statement**.
- **distinct** – allows duplicate rows; and if the original definition of the view did not specify `distinct` clause, you can change this parameter so the new view cannot contain duplicate rows.
- *select_statement* – specify different tables and views in the definition of the view. The columns specified in the target list of the **select_statement** of the replaced view can be changed to drop or add columns.
- **with check option** – a view created with the **with check option** clause can be replaced without this clause, and vice versa.

Examples

Example 1

This example is based on the view `Current_Product_List` which lists all active products from the table `Products`. The view is defined as:

```
create view Current_Product_List as
select ProductID, ProductName
from Products
where Discontinued = "No"

select object_id("Current_Product_List")
-----------
889051172
```

This next command adds the `Category` column to `Current_Product_list` using the **or replace** clause. The object ID of the view remains the same:

```
create or replace view Current_Product_List as
select ProductID, ProductName, Category
from Products
where Discontinued = "No"

select object_id("Current_Product_List")
-----------
889051172
```

Example 2

In this example, V1—a view that has dependent objects—is replaced:

```
create table T1(C1 int, C2 int)
create table T2(C1 int, C2 int)

create view V1 as select * from T1
create view V2 as select * from V1

create function foo1
returns int
```

```
as
begin
    declare @number int
    select @number = C1 from V2
end
return @number
select object_id("V1")
-----------
985051514
```

```
create or replace V1 as select * from T2

select * from V2

select dbo.foo1()

select object_id("V1")
-----------
985051514
```

The replaced version of `V1` references `T2` instead of `T1`. Both `V2` and `foo1` will be recompiled. `select * from v2` recompiles `V2`, but not `foo1`, which is recompiled when the UDF is invoked.

Objects Dependent on Replaced Views

Views can be contained in other object definitions.

- If the view that is replaced is contained in another view and the parent view is automatically recompiled when it is accessed.
- If the number of columns in a view changes due to replacing, you may need to fix the definitions of other views and procedures that reference this view.

  In this situation, the owner has replaced `V1` with different number of columns and column names. `P` must also be replaced.

  ```
  create view V1 as select C1 from T1
  create procedure P as select * from V1
  ```

  ```
  create or replace V1 as select C1, C2 from T1
  ```

  In this next situation, the owner has replaced V2 by removing C2 from the definition. When P is executed, an error is raised, as C2 is no longer part of V2 . Because of this, P must be replaced.

  ```
  create view V2 as select C1, C2 from T2
  create procedure P as select C2 from V2
  ```

  ```
  create or replace V2 as select C1 from T2
  ```

  Before you replace a view, run **sp_depends** to determine if there are any stored procedures or parent views that depend on the view you are replacing. If such stored procedures or parent views exist, replace the stored procedures or parent views as necessary after replacing the view.

- `Instead of` triggers defined on the view are dropped when the view is replaced
- Any PRS that are dependent on the replaced view will require a full refresh to restore them to a usable state. You can neither refresh them, nor use them for query rewrite until they are recompiled.

Restrictions

If a view has permissions granted at the column level, then it cannot be replaced and error 2014 is raised. To replace the view, you must first revoke the permissions, or drop and re-create the view.

*Permission Changes for **create or replace view***

Any user who impersonates the view owner through an alias or **setuser** cannot replace the view.

Changes for replacing a view are in bold.

Granular permissions enabled	With granular permissions enabled, you must have the `create view` privilege. You must have the `create any view` privilege to run **create view** for other users. **You must be the view owner to replace the view.**
Granular permissions disabled	With granular permissions disabled, you must be the database owner or have the `create view` privilege. **You must be the view owner to replace the view.**

*Auditing Changes for **create or replace view***

Changes are in bold.

Event	Audit Option	Command or access audited	Information in extra-info
13	create	create view	• Roles – current active roles • Keywords or options – NULL • Previous value – NULL • Other information – NULL • Current value – NULL • Proxy information – original login name, if set proxy is in effect • **or replace – for create or replace**

drop encryption key

Deletes the database encryption key from the `sysencryptkeys` table in the `master` database.

Syntax

```
drop encryption key key_name
```

Parameters

- ***key_name* –** is the name of the database encryption key.

Usage

This command fails if the database encryption key you are dropping is still used to encrypt any database.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

The permission checks for **drop encryption key** differ, based on your granular permission settings:

Granular permissions enabled	SAP ASE creates a new permission called "`manage database encryption key`." You must have this permission to create a database encryption key.
Granular permissions disabled	You must be a user with sso_role, keycustodian_role, or have **create encryption key** privilege.

See also

- *create encryption key* on page 122
- *Changing a Database Encryption Key* on page 67
- *Dropping a Database Encryption Key* on page 68
- *Back Up the Database Encryption Key* on page 78
- *Creating the Database Encryption Key* on page 65
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177

drop trigger

You can use **drop trigger** to remove or replace an existing trigger, or multiple triggers.

SAP ASE version 16.0 introduces the ability to create multiple triggers, as well as specify the order in which the triggers are fired after statement execution.

The **drop trigger** command drops a single trigger. If you have multiple triggers on a table, you can drop them individually.

dump database

dump database adds a cyclic redundancy check for accidental changes to raw data for database or transaction dumps created with compression to check and for verification that the compression blocks can be correctly read and decompressed

The syntax is:

```
dump database database_name to dump_device with
compression=n,verify={crc | read_after_write}
```

Where:

- **verify=crc** – indicates that you are performing a cyclic redundancy check.
- **verify=read_after_write** – Backup Server rereads every compressed block after writing and decompressing it. If Backup Server finds an error, it prints the offset in the file it finds the error. You cannot use **verify=read_after_write** with **load database** commands.

kill

The **kill** command adds the **with force** parameter.

Use the **with force** parameter if you cannot terminiate the process with the regular **kill** *spid* parameter.

Syntax

The syntax is:

```
kill spid with force
```

where **with force** indicates you are forcibly terminating the indicated spid.

Example

This example terminates spid 16:

```
kill 16 with force
```

Usage

- SAP ASE issues this message if you use **with force** to terminate a spid that holds spinlocks:

  ```
  You cannot kill spid 'spid_number' with force option as it is
  holding spinlock(s).
  ```
- The permissions for issuing the **with force** parameter are the same as for issuing **kill** *spid_number*.
- *spid_number* must be a constant; it cannot be passed as a parameter to a stored procedure or used as a local variable.

load database

load database adds a cyclic redundancy check for accidental changes to raw data for compressed database or transaction dumps.

The syntax is:

```
load database database_name from dump_device with verify[only]=crc
```

Where:

- **verify=crc** – indicates that you are performing a cyclic redundancy check.

select

Issuing **select** statements that reference only @variables, @@global variables, and constants on SAP ASE version 16.0 and later in chained mode do not start new transactions.

This enables you to create SQL constructs similar to:

```
set chained ON
<... Perform some DML or other commands ...>
select @@error, @@trancount, @@transtate
```

In some circumstances, SAP ASE may rollback the active transaction if an error occurs while executing DML or other statements. Use the **select** statement to check the state of the transaction while the server is in chained mode, and—without automatically starting a new transaction—collect the date and time the error occurred that triggered the rollback.

Although most functions start new transactions when used in **select** statements, these diagnostic functions do not:

- **getdate**
- **getutcdate**
- **current_date**
- **current_time**
- **current_bigdatetime**
- **current_bigtime**
- **abs**
- **asehostname**
- **hostname**
- **switchprop**

select into

The **select into** syntax has been extended to create an index compressed table by selecting from an existing table.

Syntax

```
select [ all | distinct ] select_list
    into [[database.][owner].table_name
    [with {max_rows_per_page = num_rows,
        exp_row_size = num_bytes,
        reservepagegap = num_pages,
        identity_gap = value,
        transfer table [on | off],
        compression [={NONE | PAGE | ROW | ALL} ] ,
        index_compression [={NONE | PAGE} ]
        }
    ]
    [on segment_name]
```

Parameter Changes

- NONE – indexes on the specified table are not compressed. Indexes that are specifically created with **index_compression = PAGE** are compressed.
- PAGE – all indexes on the specified table are compressed. Indexes that are specifically created with **index_compression = NONE** are not compressed.

See also

- *Creating an Index Compressed Table* on page 58
- *create table for Index Compression* on page 140

set

SAP ASE version 16.0 introduces changes to the **set** command.

set spinlock_aggregation {on | off}

Use the **set spinlock_aggregation** parameter to determine whether SAP ASE aggregates the spinlock metrics reported in `monSpinlockActivity` when the result set includes multiple rows with the same value for `SpinlockName`.

By default, SAP ASE aggregates the values for `Grabs`, `Waits`, and `Spins` for all spinlocks with the same value. Set **set spinlock_aggregation** to off to configure SAP ASE to return a separate row for each spinlock instance in the `monSpinlockActivity` table.

This example shows a result set from `monSpinlockActivity` with **set spinlock_aggregation** disabled:

```
1> set spinlock_aggregation off
2> go
1> select * from monSpinlockActivity
2> where SpinlockName like "default data cache%"
3> order by Contention
4> go
Grabs               Spins               Waits               OwnerPID
        LastOwnerPID Contention                  InstanceID
SpinlockSlotID
        SpinlockName
 -------------------- -------------------- --------------------
-----------
        ------------ -------------------------- ----------
--------------

-----------------------------------------------------------------------
---
              37697                978                   1          0
           1638413                  0.000027         0          2338
       default data cache                16396                15
             2           0
           1638413                  0.000122         0          2340
        default data cache
             17317                 24                   3          0
           1638413                  0.000173         0          2339
        default data cache
             27533                629                   5          0
           1638413                  0.000182         0          2341
        default data cache
```

When you select the number of rows from this instance of `monSpinlockActivity`:

```
select count(*) from monSpinlockActivity
-----------
        2384
```

However, if you enable **set spinlock_aggregation** and perform the same query:

```
1> set spinlock_aggregation on
2> go
1> select * from monSpinlockActivity
2> where SpinlockName like "default data cache%"
3> order by Contention
4> go

Grabs               Spins               Waits               OwnerPID
         LastOwnerPID Contention                  InstanceID
SpinlockSlotID
         SpinlockName
 -------------------- -------------------- --------------------
-----------
         ------------ -------------------------- ----------
--------------

-------------------------------------------------------------------
              99235               1646                 11          0
            1769486                 0.000111         0          2338
         default data cache
```

This instance of `monSpinlockActivity` now shows much fewer rows:

```
select count(*) from monSpinlockActivity
-----------
        324
```

set erase_residual_data {on | off}
SAP ASE version 16.0 introduces the ability to enable the ability to erase residual data.

Using **set** allows you to enable or disable the removal of residual data based on your needs.

When you enable the option at a session level, residual data is removed from all the page deallocations that occur during that session. This includes page deallocations of tables that have the "`erase residual data`" option turned `OFF` explicitly.

This option can be set by any user for the particular session; no special permissions are required.

set statistics query_name_html [queryname | on | off]
set statistics query_name_html helps differentiate or identify files related to the execution of same query.

set lock {wait [numsecs] | nowait | default}
SAP ASE version 16.0 adds the **default** parameter to the **set** command, which disables the current session-level lock wait settings, and instead uses the current server-wide **lock wait period** configuration parameter setting.

See also

- *Multiple Triggers* on page 89
- *Query Plan and Execution Statistics in HTML* on page 53

System Procedures

SAP ASE 16.0 includes new and changed system procedrues.

Changed System Procedures

SAP ASE 16.0 includes changes to existing system procedures.

sp_audit

SAP ASE 16.0 adds the **config_history** auditing option.

Option	Description
config_history	Enables or disables auditing for configuration history

sp_chgattribute

The **pnt_locking** table attribute enables and disables partition level locking with **sp_chgattrbute**. By default, partition locking is disabled.

Syntax

```
sp_chgattribute  objectname, 'ptn_locking', value
```

Parameters

- ***objectname* –** is the name of the table on which to change **ptn_locking**.
- **ptn_locking –** set to 1 to enable and 0 to disable partition-level locking.

Examples

- **Example 1 –** This example enables partition-level locking for the `authors` table:

  ```
  sp_chgattribute authors, "ptn_locking", 1
  ```

- **Example 2 –** This example disables partition-level locking for the `authors` table:

  ```
  sp_chgattribute authors, "ptn_locking", 0
  ```

Permissions

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

sp_clusterlockusage

sp_clusterlockusage output has been enhanced to print cluster lock usage information that is specific to partition locks.

```
Lock Usage                        count      % of total
-------------------------  ------------   ------------
Total Locks                     1479860        n/a
 Free Locks                     1469318       99.29 %
 Used Locks                      10542        0.71 %
 Object Locks                     7948        0.54 %
 Physical Locks                   1864        0.13 %
 Table Locks                         0        0.00 %
 Partition Locks                     8        0.00 %
 Page Locks                          0        0.00 %
 Row Locks                          42        0.00 %
 Others                            680          0.05 %
```

sp_dboption

The **sp_dboption** system procedure supports the ability to remove residual data from deletions in SAP ASE.

Syntax

The syntax to enable the removal of residual data at the database level is:

```
sp_dboption dbname, "erase residual data", true
```

Examples

The following examples use these two tables:

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

Example 1

The option to erase residual data is turned on for table `t1` because it is set at the database level, so that both the **drop table** and **truncate table** commands for `t1` result in the cleanup of all residual data from its pages.

Table `t2`, however, has the **erase residual data** option turned off explicitly, as it was created with the "**erase residual data off**" clause. Residual data is not removed, even though the "`erase residual data`" option is set to `true` at the database level. As a result, residual data remains, even after running **drop table** and **truncate table** on `t2`:

```
create database db1
go
sp_dboption db1, "erase residual data", true
go
use db1
go
create table t1 (col int)
go
insert t1 values ...
```

```
go
create table t2 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
drop table t2
go
```

Example 2

In this example:

- Table `t1` does not have "`erase residual data off`" set explicitly, but does have it set at the database level, resulting in the removal of residual data from `t1` when you run **truncate table t1**.
- Table `t2` has the "`erase residual data`" option set at creation, because the option was set at the database level. This results in the removal of residual data from `t2` when you run **truncate table t2**.
- Table `t3` is marked with "`erase residual data off`" explicitly, so that even though **sp_dboption** sets "`erase residual data`" to true, residual data is not removed when SAP ASE runs **truncate table t3**.

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

Usage

When you enable this setting at the database level, any operation that results in deallocation is followed by the cleaning of its pages. By default, this option is disabled.

Note: Using this option can have a major impact on performance because you do not have control at the granular level.

Permissions
To set the database-level option using **sp_dboption**, the user must be a system administrator or database owner.

See also
- *create table for Residual Data Removal* on page 142
- *set* on page 156

sp_depends
In SAP ASE version 16.0, you can use **sp_depends** to list the multiple triggers associated with a table.

There is no syntactical change to **sp_depends**. View the list of multiple triggers associated with each DML action with this syntax:

```
sp_depends table_name
```

sp_encryption
The **sp_encryption** system procedure reports a new key type called "`database encryption key`" to show if a database is fully encrypted.

For example:

```
1> create encryption key key1 as default for database encryption
2> go
1> sp_encryption helpkey, key1

Key Name    Key Owner    Key Length   Key Algorithm
    Key Type                           Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    -------------------------------   ------ ---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
    symmetric database encryption key      0                       1
  master key       0                              0
1> create encryption key key2 for database encryption with master key
2> create encryption key key3 for database encryption with
dual_control
3> go
1> sp_encryption helpkey, 'key%'
Key Name    Key Owner    Key Length   Key Algorithm
    Key Type                           Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    -------------------------------   ------ ---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
    symmetric database encryption key      0                       1
  master key       0                              0
```

```
key2        dbo          256          AES
    symmetric database encryption key      0                       1
  master key       0                              0
key3        dbo          256          AES
    symmetric database encryption key      0                       1
  dual_control(master key + dual master key)  0                  0

1> create database encr_db1 encrypt with key1
2> create database encr_db2 encrypt with key2
3> create database encr_db3 encrypt with key3
4> go
1> sp_encryption helpkey, '%', "display_dbs"
Key Name Key Owner Encrypted Database
-------------------- --------------------------------------
key1 dbo encr_db1
key1 dbo encr_db2
key3 dbo encr_db3
```

See also

- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

sp_familylock

A `partitionid` column has been added to the output of the **sp_familylock** stored procedure.

The syntax is:

```
sp_familylock
```

Table lock and fine-grained lock values for `partitionid` are 0. `partitionid` is populated only for partition-level locks.

```
spid  locktype             table_id     partitionid     page   row…
----  -------------------  -----------  -------------   -----  ----
0     Ex_partition           576002052      576004423       0     0
0     Sh_partition_intent   1417053053     1417053053       0     0
```

See also

- *View Partition Locks with sp_familylock* on page 7

sp_helpdb

The **sp_helpdb** system procedure supports the full database encryption feature.

When you run **sp_helpdb** on a fully encrypted database, it reports its encryption status:

- Encrypted
- Encryption in progress
- Decryption in progress

If the database is being encrypted or decrypted, **sp_helpdb** reports the percentage of work that has completed.

Examples

Example 1

Reports the status of database that is being encrypted:

```
>sp_helpdb
>go
name         db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db      6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    encryption in progress: 35%
……
```

Example 2

Reports the status of a partially encrypted database:

```
>sp_helpdb
>go
name         db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db      6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    encrypted partly

……
```

Example 3

Reports the status of a database that is partially decrypted:

```
>sp_helpdb
>go
name         db_size    owner dbid  created      durability
    lobcomplvl inrowlen
```

```
status
……
test_db     6.0 MB     sa        4  Aug 07, 2013 full
             0 NULL
    decrypted partly

……
```

See also

- *Encrypt an Existing Database* on page 69
- *dbencryption_status* on page 111
- *Suspend the Encryption Process* on page 75
- *Resume the Encryption Process* on page 76
- *alter database for Full Database Encryption* on page 113
- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

sp_lock

The `partitionid` column has been added to the output of the **sp_lock** stored procedure.

Table lock and fine-grained lock values for `partitionid` are 0. `partitionid` is populated only for partition-level locks.

This example shows all locks, including partition locks, currently held by SAP ASE.

```
sp_lock
go

fid spid loid locktype table_id partitionid page row dbname class
context
--- ---- ---- -------- -------- ----------- ---- --- ------ ------
-------
0 13 26 Ex_intent 420193516 0 0 0 master Non Cursor Lock
0 13 26 Ex_intent_partition  420193516 452193630 0 0 master Non
Cursor Lock
0 13 26 Ex_page 420193516  452193630 4993 0 master Non Cursor Lock
0 14 28 Ex_intent 420193516 0 0 0  master Non Cursor Lock
0 14 28 Ex_intent_partition 420193516 468193687 0 0  master Non
Cursor Lock
0 14 28 Ex_page 420193516 468193687 5001 0 master Non Cursor Lock
0 16 32 Sh_intent 1006623598 0 0 0 master Non Cursor Lock
```

See also

- *View Partition Locks with sp_lock* on page 7

New System Procedures

SAP ASE version 16.0 introduces new system procedures.

sp_confighistory

Creates the `ch_events` view and displays changes made to SAP ASE configuration.

Syntax

```
sp_confighistory create_view
    begin_date[, end_date]]
    last[items_num]
    {area | type | target | element}[, item_name]
    help
```

Parameters

- **create_view –** indicates you are creating the `ch_events` view.
- ***begin_date*, [*end_date* –** displays all items from *begin_date* value to the *end_date* value.
- **last –** displays the latest configuration history items.
- ***items_num* –** number of items to show from the list of latest configuration history items.
- **area | type | target | element –** displays items from the specified area:
  - **area** – area in which the auditable event occurs. One of:
    - `server` – server-level events.
    - `database` – database-level events.
    - `cache` – cache-level events.
    - `traceflag` – **dbcc traceflag** and **set switch** events.
    - `SUSD` – startup/shutdown.
    - `audit` – auditing state changes.
  - **type** – type of auditable event. One of:
    - **sp_configure**
    - **sp_serveroption**
    - **sp_dboption**
    - **sp_cacheconfig**
    - **sp_poolconfig**
    - **create thread pool**
    - **alter thread pool**
    - **drop thread pool**
    - **dbcc traceflag**

    - **set switch**
    - configuration file change
    - startup
    - shutdown
    - shutdown with wait
    - shutdown with nowait
    - abrupt shutdown
    - global auditing
    - config history auditing
  - **target** – name of the target objects to which the change applies (for example, server, cache, thread pool, and database names, traceflag number, and so on).
  - **element** – configuration or other option name (for example, "`enable monitoring`", "`config pool: 4K, option: wash size`", and so on).
- **help –** displays usage information for **sp_confighistory**.

Examples

-

Permissions

- Only the system administrator (users with sa_role) can use this procedure to create the `ch_events` view.
- Only the system administrator (users with sa_role) and users with mon_role can use this procedure to query the `ch_events` view.

The permission checks differ, based on your granular permission settings:

Setting	Description
Enabled	Only users with: • `select any audit table` permission can query against the `ch_events` view. • `manage auditing` permission can change the option state of configuration history auditing • `select any audit table` permission can query against the `ch_events` view. • `select any audit table` permission can query the audit tables.

Setting	Description
Disabled	Only: • System security officers (users with sso_role) can change the option state of configuration history auditing • only system administrators (users with sa_role) and users with mon_role can query against the ch_events view.

sp_ dropglockpromote_ptn

Removes partition lock promotion values.

Syntax

The syntax for dropping server-wide partition lock promotion settings is:

```
sp_dropglockpromote_ptn "server"
```

The syntax for dropping the partition lock promotion threshold at the database or table level is:

```
sp_dropglockpromote_ptn {"database" | "table"}, objname
```

Parameters

- **server –** removes server-wide values for the partition lock promotion thresholds.
- **"database" | "table" –** specifies whether to remove the partition lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is the name of the table or database from which to remove the partition lock promotion thresholds.

Examples

- **Example 1 –** Removes the partition lock promotion values from *titles*. Lock promotion for *titles* now uses the database or server-wide values:

  ```
  sp_ dropglockpromote_ptn "table", titles
  ```

Usage

There are additional considerations when using **sp_dropglockpromote_ptn**:

- Use **sp_dropglockpromote_ptn** to drop partition lock promotion values set with **sp_setpglockpromote_ptn**.
- When you drop a database's partition lock promotion thresholds, tables that do not have partition lock promotion thresholds configured use the server-wide values.
- When a table's values are dropped, the SAP ASE server uses the database's lock promotion thresholds if they are configured or the server-wide values if they are not.

- When you drop server-wide partition lock promotion thresholds, partition lock promotion threshold values set at the table level will be used. Otherwise, partition lock promotion threshold values set at the database level will be used. If partition lock promotion threshold values are not set at either database or table level, then partition lock promotion is disabled. It can be enabled again using **sp_setrowlockpromote_ptn**.

Permissions

The permission checks for **sp_dropglockpromote_ptn** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_droprowlockpromote_ptn* on page 168
- *Dropping Partition Lock Promotion Thresholds* on page 9

sp_droprowlockpromote_ptn

Removes partition lock promotion threshold values at server, database, or table levels.

Syntax

The syntax for dropping server-wide partition lock promotion settings is:

```
sp_droprowlockpromote_ptn "server"
```

The syntax for dropping the partition lock promotion threshold at the database or table level is:

```
sp_droprowlockpromote_ptn {"database" | "table"}, objname
```

Parameters

- **server –** removes server-wide values for the partition lock promotion thresholds.
- **"database" | "table" –** specifies whether to remove the partition lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is the name of the table or database from which to remove the partition lock promotion thresholds.

Examples

- **Example 1 –** Removes the partition lock promotion values from the `sales` table. Partition lock promotion for `sales` now uses the database or server-wide values:

  ```
  sp_droprowlockpromote_ptn "table", "sales"
  ```

Usage

There are additional considerations when using **sp_droprowlockpromote_ptn**:

- Use **sp_droprowlockpromote_ptn** to drop partition lock promotion values set with **sp_setrowlockpromote_ptn**.
- When you drop a database's partition lock promotion thresholds, datarows-locked tables that do not have partition lock promotion thresholds configured at table level use the server-wide values. Use **sp_configure** to check the value of the partition lock promotion configuration parameters.
- When a table's partition lock promotion values are dropped, the SAP ASE server uses the database's partition lock promotion thresholds, if they are configured, or the server-wide values, if no thresholds are set for the database.
- To change the partition lock promotion thresholds for a database, you must be using the `master` database. To change the partition lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- When you drop server-wide partition lock promotion thresholds, partition lock promotion threshold values set at the table level will be used. Otherwise, partition lock promotion threshold values set at the database level will be used. If partition lock promotion threshold values are not set at either database or table level, then partition lock promotion is disabled. It can be enabled again using **sp_setrowlockpromote_ptn**.

Permissions

The permission checks for **sp_droprowlockpromote_ptn** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_ dropglockpromote_ptn* on page 167
- *Dropping Partition Lock Promotion Thresholds* on page 9

sp_helptrigger

SAP ASE version 16.0 includes the sp_helptrigger system procedure.

sp_helptrigger lists

- All triggers created on the table specified by *tablename*
- Which command (**insert**, **update**, or **delete**) fires the trigger
- The trigger's order number

Syntax

```
sp_helptrigger tablename
```

Parameters

- ***tablename* –** is the name of the table.

Permissions
Any user can execute **sp_helptrigger**.

sp_jsconfigure
Configures the Job Scheduler Agent.

Syntax
```
sp_jsconfigure [option [, value]]
```

Parameters

- ***option* –** option is one of:
  - *interfaces path* – path to the interface file
  - *errorlog* – path to the errorlog
  - *help* – displays the syntax for **sp_jsconfigure**
- ***value* –** Specifies the value to which you are setting *option*.

Examples

- **Example 1 –** Displays the **sp_jsconfigure** syntax:
```
sp_jsconfigure "help"
Usage: sp_jsconfigure [option [, value]]
        where option : 'interfaces path', ' errorlog ', 'help'
              value  : value to set for the 'option'
```
- **Example 2 –** Sets the path to the interfaces file:
```
sp_jsconfigure 'interfaces path', "/SAP_ASE/data"
```
- **Example 3 –** Sets the path to the errorlog:
```
1> sp_jsconfigure "errorlog", "/SAP_ASE/data/js.log"
```
- **Example 4 –** Displays the values to which you have set **sp_jsconfigure**:
```
sp_jsconfigure
Parameter Name                   Config
Value

 -------------------------------
-------------------------------------------------------------------
----------------------------------------
 interfaces path                  /SAP_ASE/
data
 errorlog                         /SAP_ASE/data/
js.log
```

Usage

- Use the `installjsdb` script to install **sp_jsconfigure** (located in `$SYBASE/$SYBASE_ASE/scripts`).
- You must restart JS Agent for the configuration changes to take affect.
- **sp_jsconfigure** uses default values if you do not supply values for *interfaces path* or *errorlog*.
- Job Scheduler does not start if you include paths for the *interfaces path* or *errorlog* options that do no exist.
- *interfaces path* must include an interfaces file with an entry for the Job Scheduler host and target server on which the scheduled job is executed (the target and host server can be the same machine).

Permissions

You must have the `js_admin_role` to execute **sp_jsconfigure**.

sp_logging_rate

Calculates the transaction log growth rate for the specified time period.

Syntax

```
sp_logging_rate {'full'|'sum', '[day,]hh:mm:ss'}[,
interval='hh:mm:ss' | clear_option='y'|'n']
```

Parameters

- **full – sp_logging_rate** provides a detailed report for each collection.
- **sum – sp_logging_rate** provides summary information, including values for the average, minimum, maximum, and the maximum rate. If you do not specify a time, **sp_logging_rate** collects information every 10 seconds.
- **day, *hh:mm:ss* –** Specifies the duration of time **sp_logging_rate** runs, using the form *date, hour:minute:second*.
- **interval = '*hh:mm:ss*' –** Period of time during which the interval runs, using the form *hour:minute:second*
- **clear_option = 'y' | 'n" –** Determines whether to clear the monitor counters during data collection.

Examples

- **Example using sum parameter – sp_logging_rate** collects information for 1 day and 8 hours, takes a sample every 10 minutes, and prints summary information at the end of the interval:

  ```
  sp_logging_rate 'sum', '1,08:00:00', '00:10:00'
  ========================
  ```

```
Total Summary Information
=========================
Transaction Log Growth Rate     Min GB/h       Max GB/h       Avg
GB/h
----------------------------   --------------  --------------
--------------
                                  0.000000        2.870076
1.823028
```

- **Example using full parameter – sp_logging_rate** collects information for 3 minutes, takes samples every 10 seconds (the default), and prints summary information at the end of the interval:

```
sp_logging_rate 'full', '00:03:00'
Date Time                       Transaction Log Growth Rate GB/h
-----------------------------   --------------------------------
Oct 22 2013  6:00:32:480AM                0.406779

Oct 22 2013  6:00:42:483AM                0.000000

Oct 22 2013  6:00:52:483AM                0.000000

Oct 22 2013  6:01:02:483AM                0.000000

Oct 22 2013  6:01:12:490AM                0.000000

Oct 22 2013  6:01:22:500AM                0.000000

Oct 22 2013  6:01:32:476AM                2.341870

Oct 22 2013  6:01:42:483AM                2.828132

Oct 22 2013  6:01:52:480AM                2.850305

Oct 22 2013  6:02:02:483AM                2.782750

Oct 22 2013  6:02:12:483AM                2.853574

Oct 22 2013  6:02:22:480AM                2.002917

Oct 22 2013  6:02:32:483AM                2.848995

Oct 22 2013  6:02:42:483AM                2.754143
```

```
Oct 22 2013  6:02:52:483AM                 2.854949

Oct 22 2013  6:03:02:480AM                 2.722928

Oct 22 2013  6:03:12:476AM                 2.870076

Oct 22 2013  6:03:22:480AM                 2.697094


=========================
Total Summary Information
=========================
Transaction Log Growth Rate     Min GB/h       Max GB/h       Avg
GB/h
----------------------------   -------------  -------------
--------------
                                   0.000000        2.870076
1.823028
```

Usage

- You cannot run scripts or procedures that collect monitoring data (for example, sp_sysmon) while **sp_logging_rate** runs. Because **sp_logging_rate** collects and clears monitor counter as it runs, the monitoring counter information these scripts or procedures collect will not be accurate.
- **sp_logging_rate** produces unreliable results if you specify an amount of time for **interval =** '*hh:mm:ss*' that is greater than the amount of time you specify for **'day,** *hh:mm:ss*'.
- When you specify values for **interval =** '*hh:mm:ss'* and **'day,** *hh:mm:ss'*, keep in mind:
  - If the value you specify for **interval =** '*hh:mm:ss*' is greater than the value you specify for **'day,** *hh:mm:ss'*, SAP ASE issues an error message and **sp_logging_rate** produces no result set.
  - **sp_logging_rate** may produce an unreliable result if that ratio for **'day,** *hh:mm:ss'* to **interval =** '*hh:mm:ss'* is too small. For example, if you specify `day,00:10:00,` and `interval='00:04:00'`, **sp_logging_rate** collects only two values, prints an average value, with the first value as the maximum, and the second value as the minimum. A better ratio produces a more reliable result set.

Permissions

You must have system administrator privileges to execute **sp_logging_rate**.

sp_setpglockpromote_ptn

The **sp_setpglockpromote_ptn** system procedure sets partition-lock promotion thresholds at the server, database, and table level.

Syntax

The syntax for setting the partition lock promotion threshold at the server level is:

```
sp_setpglockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

The syntax for setting the partition lock promotion threshold at the database or table level is:

```
sp_setpglockpromote_ptn "database | table", objname, new_lwm,
new_hwm, new_pct
```

Parameters

- **server –** sets server-wide values for the lock promotion thresholds.
- **"database" | "table" –** specifies whether to set the lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is either the name of the partition, table, or database for which you are setting the lock promotion thresholds, or null, if you are setting server-wide values. If you are setting partition-wide values, use the format *table_name.partition_name* for the *objname*.
- ***new_lwm* –** specifies a minimum number of page locks that must be acquired before SAP ASE acquires a partition lock.
- ***new_hwm* –** specifies a maximum number of page locks allowed on the object before SAP ASE attempts to escalate to a partition lock.
- ***new_pct* –** specifies the percentage of locked pages (based on the table size) above which SAP ASE attempts to acquire a partition lock when the number of locks is between the *new_hwm* and *new_lwm* lock promotions.

Examples

- **Example 1 –** Sets the server-wide partition lock promotion threshold values LWM to 200, the HWM to 300, and the PCT to 50:

  ```
  sp_setpglockpromote_ptn "server", NULL, 200, 300, 50
  ```

- **Example 2 –** Sets partition lock promotion thresholds for the master database:

  ```
  sp_setpglockpromote_ptn "database", master, 1000, 1100, 45
  ```

- **Example 3 –** Sets partition lock promotion thresholds for the titles table in the pubs2 database. This command must be issued from the pubs2 database:

  ```
  sp_setpglockpromote_ptn "table", "pubs2..titles", 500, 700, 10
  ```

Permissions

Any user can execute **sp_setpglockpromote_ptn**.

See also

- *sp_setrowlockpromote_ptn* on page 176
- *Setting Partition Lock Promotion Thresholds* on page 9

sp_setrowlockpromote_ptn

The **sp_setrowlockpromote_ptn** system procedure sets partition-lock promotion thresholds at the server, database, and table level.

Syntax

The syntax for setting the partition lock promotion threshold at the server level is:

```
sp_setrowlockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

The syntax for setting the partition lock promotion threshold at the database or table level is:

```
sp_setrowlockpromote_ptn "database | table", objname, new_lwm,
new_hwm, new_pct
```

Parameters

- **server –** sets server-wide values for the lock promotion thresholds.
- **"database" | "table" –** specifies whether to set the lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is either the name of the partition, table, or database for which you are setting the lock promotion thresholds, or null, if you are setting server-wide values. If you are setting partition-wide values, use the format *table_name.partition_name* for the *objname*.
- ***new_lwm* –** specifies a minimum number of row locks that must be acquired before SAP ASE acquires a partition lock.
- ***new_hwm* –** specifies a maximum number of row locks allowed on the object before SAP ASE attempts to escalate to a partition lock.
- ***new_pct* –** specifies the percentage of locked rows (based on the table size) above which SAP ASE attempts to acquire a partition lock when the number of locks is between the *new_hwm* and *new_lwm* lock promotions.

Examples

- **Example 1 –** Sets the server-wide partition lock promotion threshold values LWM to 200, the HWM to 300, and the PCT to 50:

  ```
  sp_setrowlockpromote_ptn "server", NULL, 200, 300, 50
  ```

- **Example 2 –** Sets partition lock promotion thresholds for the master database:

  ```
  sp_setrowlockpromote_ptn "database", master, 1000, 1100, 45
  ```

- **Example 3 –** Sets partition lock promotion thresholds for the titles table in the pubs2 database. This command must be issued from the pubs2 database:

```
sp_setrowlockpromote_ptn "table", "pubs2..titles", 500, 700, 10
```

Permissions

Any user can execute **sp_setrowlockpromote_ptn**.

See also

- *sp_setpglockpromote_ptn* on page 175
- *Setting Partition Lock Promotion Thresholds* on page 9

System Tables

SAP ASE 16.0 includes new and changed tables.

Changed System Tables

SAP ASE version 16.0 introduces changes to system tables.

sysattributes

The `sysattributes` system table supports the full database encryption feature. The encrypted database feature introduces 43, a new `systtributes` class that signifies full database encryption. For every storage allocation of the database that undergoes encryption, SAP ASE inserts a row in `sysattributes` with these values:

Column Name	Value
class	43
object	*dbid* (database ID)
object_info1	Starting logical page ID
object_info2	Ending logical page ID
int_value	Last encrypted logical page ID on one storage allocation

This row is removed when SAP ASE finishes encrypting the database.

sysconstraints

In versions of SAP ASE earlier than 16.0, `sysconstraints` saved information about check constraints, unique and primary key constraints, referential constraints, and rules and computed column definitions. In SAP ASE 16.0 `sysconstraints` also stores the following information about multiple triggers associated with a table:

Name	Datatype	Description
colid	smallint	Ordering number of trigger. The default is 0.
constrid	int	Trigger ID.
tableid	int	ID of the table on which the trigger is declared.
error	int	Not used for triggers.
status	int	0x0080 = a **delete** trigger 0x0100 = an **insert** trigger 0x0200 = an **update** trigger 0x0400 = trigger is disabled
spare	int	Unused

sysdatabases

The sysdatabases system table supports the full database encryption feature. The encrypted database feature introduces status5, a new column that indicates the encryption status of a database. The values are:

Hex	Description
0x00000001	Indicates whether the database is encrypted or not.
0x00000002	The database is being encrypted, and the encryption is still in progress.
0x00000004	The database is being decrypted, and the decryption is still in progress.
0x00000008	The database is only partially encrypted, either due to an error or because the process was suspended by the user.
0x00000010	The database is only partially decrypted, either due to an error or because the process was suspended by the user.

sysobjects

In versions of SAP ASE earlier than 16.0, sysobjects saved the following:

- deltrig column for **delete** triggers
- instrig column for **insert** triggers
- updtrig column for **update** triggers

In SAP ASE 16.0, the first trigger created on a table for **delete**, **insert**, and **update** operations, where the trigger is created without the order clause (or order 0) is associated with the table in one of the above columns in sysobjects.

The second and subsequent triggers created for a given action are associated with the table through a row in `syscontraints`. In addition, any trigger with `order 1` or greater always uses `sysconstraints` for the table/trigger association.

The `sysobjects` row for the dependent table uses bits in the `sysstat2` field to indicate that a trigger is disabled. In versions earlier than SAP ASE 16.0, there could be no more than one insert, delete, or upgrade trigger, which used these three bits:

- `disable_instrig` 0x001000000
- `disable_deltrig` 0x002000000
- `disable_updtrig` 0x004000000

These bits exist in SAP ASE 16.0 and are used when a table's trigger ID is stored in `sysobjects`, as described above.

See also

- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *sybmigrate* on page 187
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

New System Tables

SAP ASE version 16.0 introduces new system tables.

ch_events

Contains one row for each configuration change event. `ch_events` is located in the `sysmgmtdb` database.

`ch_events` is a view based on the `extrainfo` columns. You must have the mon_role to view `ch_events`.

Columns

The columns for `ch_events` are:

Name	Datatype	Description
area	varchar(10) not null	Area in which the event occurs. One of: • server – server-level events. • database – database-level events. • cache – cache-level events. • traceflag – **dbcc traceflag** and **set switch** events. • SUSD – for startup/shutdown. • audit – auditing state changes.
type	varchar(30) not null	Type of auditable event. One of: • sp_configure • sp_serveroption • sp_dboption • sp_cacheconfig • sp_poolconfig • create thread pool • alter thread pool • drop thread pool • dbcc traceflag • set switch • configuration file change • startup • shutdown • shutdown with wait • shutdown with nowait • abrupt shutdown • global auditing • config history auditing .
target	varchar(30) null	Name of the objects to which the change applies.
element	varchar(255) null	Configuration parameter or other option name.
oldvalue	varchar(255) null	Value of event prior to change.

Name	Datatype	Description
newvalue	varchar(255) null	Value of event after change.
mode	varchar(10) null	Status for configuration parameters: static or dynamic.
timestamp	datetime not null	Date and time the event takes place. For changes to the configuration file and abrupt shutdowns, timestamp indicates the time the event was detected, not when the event took place.
username	varchar(30) null	Name of the user who made the change. Set to null for: • Startup • Configuration file change • Abrupt shutdown
instanceid	tinyint null	(Cluster Edition only) ID of the instance.

Changed Monitoring Tables

SAP ASE 16.0 includes changes to existing monitoring tables.

monCachedStatement

monCachedStatement in SAP ASE 16.0 and later:

- Updates the metrics for these columns approximately every 5 seconds for completed and in-process queries:

Description	Datatype	Attribute	Description
TotalLIO	bigint	counter	Cumulative logical I/O
TotalPIO	bigint	counter	Cumulative physical I/O
TotalCPUTime	bigint	counter	Cumulative elapsed time, in seconds, this statement spent using CPU
TotalElapsedTime	bigint	counter	Cumulative amount of time, in seconds spent executing this statement

  Versions of SAP ASE earlier than 16.0 updated metrics in monCachedStatement when the statement finished. However, when SAP ASE 16.0 and later executes a statement cache, it periodically updates these values while it executes a query:
  - TotalLIO

- MaxLIO
- TotalPIO
- MaxPIO
- TotalCPUTime
- MaxCPUTime
- TotalElapsedTime
- MaxElapsedTime

Other metrics (for example, MinLIO and AvgLIO) are updated after query executions are finished.

- Increments the UseCount column when statement begin execution. The value for UseCount is

```
(number of completed queries) + (number of ongoing queries)
```

The CurrentUsageCount column includes the number of active queries for a statement. The number of completed executions for a statement is:

```
(Value of UseCount) – (value of CurrentUsageCount)
```

- Increments the value for columns that describe maximums (for example, MaxCPUTime) for currently executing statements if the metric described by the column (in this case, CpuTime) exceeds the maximum value used during an intermediate update. Maximum columns reflect up-to-date metrics (including metrics for active queries), which helps determine if a currently executing query is consuming resources that exceed previous or normal usage.

monDeadLock

Description	Datatype	Attribute	Description
partitionid	int	Null	Unique identifier for the partition

monLocks

Description	Datatype	Attribute	Description
partitionid	int	Null	Unique identifier for the partition

monOpenObjectActivity

SAP ASE version 16.0 and later adds these columns to the monOpenObjectActivity monitoring tables.

Description	Datatype	Attribute	Description
Scans	int	counter	Number of scans performed on this object.
LastScanDate	datetime	counter	Date of the last scan on this object

Description	Datatype	Attribute	Description
Updates	int	counter	Number of updates performed on this object.
LastUpdateDate	datetime	counter	Date of the last update on this object
Inserts	int	counter	Number of inserts performed on this object.
LastInsertDate	datetime	counter	Date of the last insert on this object
Deletes	int	counter	Number of deletes performed on this object.
LastDeleteDate	datetime	counter	Date of the last delete on this object

monOpenPartitionActivity

SAP ASE version 16.0 and later adds these columns to the `monOpenPartitionActivity` monitoring tables.

Description	Datatype	Attribute	Description
Scans	int	counter	Number of scans performed on this object.
LastScanDate	datetime	counter	Date of the last scan on this object
Updates	int	counter	Number of updates performed on this object.
LastUpdateDate	datetime	counter	Date of the last update on this object
Inserts	int	counter	Number of inserts performed on this object.
LastInsertDate	datetime	counter	Date of the last insert on this object
Deletes	int	counter	Number of deletes performed on this object.
LastDeleteDate	datetime	counter	Date of the last delete on this object

monProcess

Description	Datatype	Attribute	Description
ClientDriverVersion	varchar16		Version of the connectivity driver used by the client program

monRepLogActivity, monRepScannersTotalTime, and monRepSenders

SAP ASE versions 16.0 and later require that you enable the **activate monitoring** configuration parameter for the `monRepLogActivity`, `monRepScannersTotalTime`, and `monRepSenders` monitoring tables to start collecting monitoring data.

monRepScanners
SAP ASE version version 16.0 makes these changes to `Status` column of the `monRepScanners` monitoring table:

- Removes the value `spawned`
- Adds the values `sync mode`, `async mode`, and `near sync mode`

monRepScannersTotalTime
SAP ASE version version 16.0 changes the name of the `MRPBootstrapTime` column to `BootstrapTime`.

monSysExecutionTime
SAP ASE version 16.0 and later include these values for the `OperationName` column:

- NetworkIO – Reports time spent sending and receiving netword data
- DeviceIO – Reports time spent performing disk IO operations
- CIPCIO – Reports time spent performing cluster interconnect network operations (Cluster Edition only)

monTables
The `Description` column in SAP ASE 16.0 and later supports 512 characters. Previous releases supported 255 characters.

monTableColumns

- The `Description` column in SAP ASE version 16.0 and later supports 512 characters. Previous versions supported 255 characters.
- The `Label` column in SAP ASE version 16.0 and later supports 150 characters. Previous versions support 50 characters.

New Monitoring Tables

SAP ASE 16.0 adds two new monitoring tables.

monThresholdEvent

The `monThresholdEvent` monitoring table includes one row for each event recorded by SAP ASE.

Enable the **allow resource limits** configuration parameter to enable resource limits collection. Enable the **enable monitoring**, **threshold event monitoring**, and **set threshold event max messages** configuration parameters for this monitoring table to collect data.

`monThresholdEvent` is a stateful historical monitoring table (see the *Performance and Tuning Guide: Monitoring Tables*). Determine the number of events `monThresholdEvent` stores with the **threshold event max messages** configuration parameter.

Columns

Name	Datatype	Attribute	Description
SPID	int4		Server process ID.
InstanceID	int1		ID of the instance within the cluster..
KPID	int4		SAP ASE kernel process ID.
KTID	int4		ID of the kernel task.
ServerUserID	int4		Server user identifier (SUID) of the user who executed this SQL text. The ServerUserID matches the value in syslogins.suid. Use the **suser_name** function to obtain the corresponding name.
FamilyID	int4	NULL	spid of the parent process.
Login	varchar(30)	NULL	Login user name.
Application	varchar(30)	NULL	Application name.
HostName	varchar(30)	NULL	Client host name.
ClientName	varchar(30)	NULL	Client name set with **set clientname**.
ClientHostName	varchar(30)	NULL	Value of the **clienthostname** property set by the application.
ClientApplName	varchar(30)	NULL	Value of the **clientapplname** property set by the application.
ClientIP	varchar(64)	NULL	IP address of the client.
Command	varchar(30)	NULL	Category of process or command the process is currently executing.
DBID	int4		Unique identifier for the database currently being used by the process.
DBName	int4	NULL	Name of the database running the process.
ProcedureID	int4		Unique identifier for the procedure.
BatchID	int4		Unique identifier for the SQL batch containing the statement being executed.
LineNumber	int4		Line number of the current statement within the SQL batch.

Name	Datatype	Attribute	Description
BlockingSPID	int4	NULL	Session process identifier of the process holding the lock this process requested, if waiting for a lock.
TempDbObjects	int4	Counter	Total number of temporary tables created by the process.
RangeID	int2		Range ID of the limit.
LimitType	varchar(30)		Limit type.
LimitID	int2		Limit identifier.
LimitValue	int4		Value of the limit that was voilated.
Enforced	int1		Determines if the limit is enforced prior to, or during, query execution.
Action	int1		Action to perform when the limit is exceeded.
Scope	int1		Scope of the limit.
ReportDatetime	datetime		Date and time the report was issued due to the limit violation.
SQLText	varchar(255)		SQL text of the event.

Utilities

SAP ASE version 16.0 introduces changes to some existing utilities.

When you install SAP ASE into a directory with a specific user, other users may not be able to use it because they lack write permission for the installed `$SYBASE` directory.

SAP ASE 16.0 lets you specify another user for both configuring new servers during or after installation, thereby allowing multiple users to use the same installed SAP ASE directory.

To support this new feature, SAP ASE 16.0 uses the `-D` *`data_directory`* option for these utilities to specify a directory other than the default.

On UNIX:

- **auditinit**
- **sqlloc[res]**
- **sqlupgrade[res]**
- **srvbuild[res]**
- **updatease**

On Windows:

- **auditinit.exe**
- **sybatch.exe**

See the individual utilities in the *Utility Guide*.

ddlgen

The **ddlgen** utility adds new functionality for SAP ASE 16.0.

ddlgen adds support for:

- Creating a fully encrypted database
- Creating a database encryption key

ddlgen adds support for transparent database encryption with these values for the **-TEK** parameter:

- -XOCE – generate DDL only from column encryption keys.
- -XOMK – generate DDL only from the master key or the dual master key.
- -XODE – generate DDL only from database encryption keys.

See also

- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *sybmigrate* on page 187
- *Changed System Tables* on page 177
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

sybmigrate

SAP ASE version 16.0 adds new functionality to the sybmigrate utility.

Including:

- You can use the **sybmigrate** utility to migrate a fully encrypted database just as you would migrate an unencrypted database. However, to encrypt the target database with the same database encryption key (DEK) as the source database, you must perform the following before migrating the source database:
  1. Use **ddlgen** to generate DDL for master key, dual master key, and the DEK (these keys are all in master database):

```
$SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
TEK -Nmaster -Dmaster -XOD
$SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
TEK -Ndualmaster -Dmaster -XOD
$SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
TEK -Ndekname -Dmaster -XOD
```

Note: You need not generate DDL for the dual master key if the DEK is encrypted only by the master key.

2. Execute the resulting DDL on the target server so it generates DEK with same raw-key value.

- You can include the **ssl** keyword with the **source_ase** or **target_ase** attribute, allowing you to connect to SSL-enabled servers. The syntax is:
  - In Resource file mode –

    ```
    [server]
    source_ase=ssl : host_name : port_number
    source_ase_login=sa
    source_ase_password=
    ```
  - In GUI mode – include "`ssl : host_name : port_number`" in the server text field

See also

- *Full Database Encryption and System Changes* on page 81
- *create archive database for Full Database Encryption* on page 119
- *dbencryption_status* on page 111
- *sp_helpdb* on page 163
- *sp_encryption* on page 161
- *ddlgen* on page 187
- *Changed System Tables* on page 177
- *alter database for Full Database Encryption* on page 113
- *create database for Full Database Encryption* on page 119
- *create encryption key* on page 122
- *drop encryption key* on page 152

sybrestore

The **sybrestore** utility supports restoring an SAP ASE server after a master database corruption.

Additional parameters have been added for interactive and noninteractive modes:

Interactive Mode

```
sybrestore
    [-J character set ]
```

```
    [-R  Restore from master database corruption ]
    [-d dump directory ]
    [-s list system databases except master database ]
    [-v version ]
    [-z language ]
    [-o Log output]
```

Noninteractive Mode

```
sybrestore
    [-o Log output]
```

See **sybrestore** in the *Utility Guide*.

Global Variables

SAP ASE version 16.0 add global variables.

Global Variable	Description
@@trig-ger_name	Returns the name of the trigger currently executing.

Global Variable	Description

@@tranroll-back	Returns the type of rollback encountered, if any. If the return value is:
	• < 0 – a server induced implicit rollback of a multistatement transaction. **@@tran-rollback** stores the negation of the error number that resulted in the implicit transaction rollback.
	• 0 – this session of the currently active transaction encountered no implicit roll-backs.
	• > 0 < 10 – the most-recent occurrence of a transaction rollback was a user-issued rollback from one of these SQL commands:
	  • **rollback tran** in a SQL batch, procedure or trigger
	  • **rollback trigger** outside a trigger's scope.
	  The return value for **@@transtate** describes which **rollback** command the user issued:
	  • 1 – user issued an explicit **rollback tran** command
	  • 2 – user issued a **rollback tran to** *savepoint*. The transaction is still active.
	• > 100 – The most recent occurrence of a transaction rollback was invoked on a single-statement transaction. **@@transtate** stores the error number that caused the statement to rollback.
	
	SAP ASE does not change a negative value for **@@tranrollback** until the next **roll-back tran** or **commit tran** is issued, indicating that the session has encountered an implicit transaction rollback. SAP ASE resets the value for **@@tranrollback** to 0 once it successfully applies the next **rollback tran** or **commit tran**. The value for **@@tranrollback** is 0 at the end of this example:
	`set chained on`
	`go`
	`<... Execute a DML statement ...>`
	`if (@@error != 0) and (@@tranrollback < 0)`
	`begin`
	`  rollback tran`
	`end`
	`go`