ADMINISTRATION GUIDE | PUBLIC
SAP Adaptive Server Enterprise 16.0 SP03
Document Version: 1.0 – 2020-03-04

Compression Users Guide

THE BEST RUN SAP

Content

1 Data Compression Overview

Data compression lets you use less storage space for the same amount of data, reduce cache memory consumption, and improve performance because of lower I/O demands.

You can compress large object (LOB) and regular data.

> **i** Note
>
> Regular data and LOB data use separate compression syntax and options. In this documentation, the phrase "data compression" indicates compression for data other than LOB columns, while "LOB compression" indicates compression for LOB columns.

SAP Adaptive Server Enterprise (SAP ASE) provides different levels of compression for regular and LOB data. Generally, higher compression ratios use more CPU when you decompress the data. Select compression levels based on how the data is accessed. Data that you access frequently ("hot data") may be best suited for compression levels that have smaller CPU overhead.

After you create a compressed table or partition, any subsequently inserted or updated data is compressed. If the inserted data cannot efficiently compressed, the original row is retained. If newly inserted or updated LOB data occupies space that is smaller than or equal to a single data page, then this data is not compressed.

Tables can be a mixture of compressed and uncompressed data. For example, if you create a compressed table, load data, then disable data compression for the table, previously inserted data is compressed, but rows added after you disable compression are not compressed.

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.

You need not uncompress data to run queries against it. You can insert, update, and delete compressed data; running `select` or `readtext` statements on the compressed column returns decompressed rows. Because there is less data to search, there are fewer I/Os, improving the efficiency of data storage.

1.1 Enabling Data Compression

To compress data, you must obtain a current ASE_COMPRESSION license, then set the system-wide configuration parameter `enable compression`.

Procedure

1. Obtain an ASE_COMPRESSION license from SAP Support Portal. See the *SySAM Users Guide* or your SAP ASE representative.

2. Enable data compression using:
   `sp_configure 'enable compression', 1`

Results

For information about `enable compression`, see *Reference Manual: Configuration Parameters*.

1.2 Selecting Data into Compressed Tables

Use `select into ... compression` to select regular and LOB data directly into a compressed table.

The destination table does not inherit anything from the original table. That is, if the table from which you are selecting data is page-level compressed, the table into which you select the data can be row-level compressed, or not compressed.

You must indicate compression levels if you are selecting large object data into a table.

The behavior of `select into` on target tables or columns depends on the type of compression you are using.

Compression Type	Behavior of `select into` on Target Tables	Source Table or Column	Database-wide Setting for `compression`	Target Table or Column
Data compression	Target table or columns do not inherit any properties from the source table. If you do not specify `compression`, tables other than temporary tables inherit the database-wide setting for `compression`. Temporary tables do not inherit any compression settings from the source table, source column, or from the target databases's attributes.	Table can be compressed or uncompressed, and may include one or more compressed columns.	`none`	Target table and all columns are uncompressed.
			`row` or `page`	Target table is created with either `row` or `page` compression, according to database-wide attribute. All eligible columns are compressed. **Note** Index compression compression does not support `row` compression.
LOB compression	LOB columns in the target table do not inherit any properties from the source col-	Source LOB columns may be compressed.	`lob_compression = 0`, unset for the database	All LOB columns in the target table are uncompressed.

Compression Type	Behavior of `select into` on Target Tables	Source Table or Column	Database-wide Setting for `compression`	Target Table or Column
	umns. If you do not specify `compression`, LOB columns in target tables other than temporary tables inherit the database-wide setting for the `lob_compression` attribute. LOB columns in temporary tables inherit nothing from the source table, source column, or from the target database's attributes.		`lob_compression = <compression_level>`	All LOB columns in the target table are created using the database-wide setting for `lob_compression = <compression_level>`.

This example selects all rows from the `titles` table, and creates a new table named `titles_2` with row-level compression:

```
select * into titles2
with compression = row
from titles
```

See the *Reference Manual: Commands*.

1.3 Administering Compressed Databases

Administration duties for compressed databases include enabling or disabling session compression, bulk-copying, and dumping and loading compressed data.

Use the:

- `compression info pool size` configuration parameter to check the memory pool for compression.
- `capture compression statistics` to enable the `monTableCompression` monitoring table to begin capturing compression statistics.

See the *System Administration Guide: Volume 1*.

1.3.1 Session-Level Data Compression

Enable and disable compression for a session with the `set` command.

To enable compression for the current session, use:

```
set compression {on | off | default}
```

This command has no effect on uncompressed tables. When you enable compression for a session, all subsequent data inserted in the table that uses the appropriate datatype is compressed. If you set

compression off, compression for the duration of the session is disabled. When you set compression to `default`, the compression configuration you established when you created the table is used.

Session-level compression for LOB compression is not supported.

Stored or system procedures inherit a session's compression settings. Subprocedures inherit the `set compression` command settings executed in the parent procedure. When the procedure ends, the compression level of the outer session or parent procedure is restored.

`set compression` changes included with login triggers apply to the session established when you first log in until you explicitly change the compression level. You need not enable `set export_options` in the login trigger to export `set compression` changes. Once the compression level is exported to a session, it applies to individual tables. However, `set compression` is not exported to the immediate parent procedure's context if you issue `set export_options` in a nested procedure before setting issuing `set compression`.

See *Reference Manual: Commands*.

1.3.2 Copy, Dump, and Load Compressed Data

Use `bcp` to bulk-copy compressed data in and out of tables.

Pages in a compressed table may have a combination of row-compressed, page-compressed, or uncompressed rows. Even tables or partitions marked as uncompressed can include data that is a mixture of different states of compression.

- `bcp out` – any compressed rows (including those with text data) are decompressed and returned to the client, either in native or character form.
- `bcp in` – uncompressed data received from the client is compressed during the insert. `bcp in` selects the appropriate compression scheme, which depends on the compression level of the partition into which you are inserting the row.

When you bulk-copy data out (using `bcp out`), followed by a `bcp in` to a compressed table (or partition), all newly loaded data is compressed, even when the extracted data was stored as uncompressed.

See *Utility Guide* > *Utility Commands Reference* > `bcp`

`dump database` dumps compressed data directly from disk to archive. If the transaction log contains compressed LOB data, recover the compressed LOB data with `load tran` (see the *System Administration Guide: Volume 2* > *Developing a Backup and Recovery Plan*).

1.4 Limits for Database Compression

Database compression includes limitations on replicating compressed data and in-memory databases.

- Generally, compression is restricted for in-memory databases. Loading and recovering compressed objects in disk-resident or relaxed-durability in-memory databases is permitted. However, access to compressed objects in the target in-memory database is often restricted. Minimal support is provided for disabling compression in the target database or in tables defined for compression, so you may revert to using uncompressed data.

- Compressed LOB columns do not support replication. Issue the following to indicate that a column is not to be replicated before you compress columns with LOB data that are part of a replicated database:

  ```
  sp_setrepcol <table_name>, <lob_column_name>, 'do_not_replicate'
  ```

  See the *Replication Server Reference Manual*.

2 Levels of Data Compression

You can compress data at the row and page level. Row-level compression compresses individual rows in a table. Page-level compression compresses data by storing repeating values or common prefixes only once.

2.1 Row-Level Compression

Row-level compression compresses individual rows in a table.

Row-level compression is intended for fixed-length, regular data. For most fixed-length columns, data does not completely occupy the space reserved for the row. For example, a 32-bit integer with a value of 2 is represented by 0x10 in hexadecimal. SAP ASE requires 1 byte to represent this value, but fills the other 3 bytes of the row with zeros. Similarly, if a 50-byte fixed-length character column includes the character data "a", SAP ASE requires 1 byte for the character data, but completes the column with space characters (0x20).

Some fixed-length datatypes are not compressed because there is no benefit in doing so. For example, SAP ASE uses only 1 byte to store a `tinyint`, so compressing a row using this datatype is not beneficial.

For example, if you create this uncompressed table:

```
create table t1 (col1 char(1) not null,
    col2 char(50) not null,
    col3 tinyint not null,
    col4 int not null,
    col5 binary(20))
lock datapages
```

After changing the compression level to `row`:

```
alter table t1
set compression = row
```

SAP ASE does not compress `col1` and `col3` because their length is 1 byte. SAP ASE compresses `col2` and `col4` and stores required information about decompression for each column using the minimum space, if required.

If you insert these values into `t1`:

```
insert t1 values (
"a", "aaaaa", 1, 100, "NineBytes")
```

The compressed version of the columns comprises 17 bytes, nearly one-third the size of the uncompressed columns:

- When uncompressed, the value of `col2`, `char(50)` is "aaaaa" with 45 blanks to fill out the rest of the column. After compression, the value of `col2` is "aaaaa", using one byte for each "a".
- The value of `col4` is 100, and is represented with a single byte.

- Trailing blanks are truncated from the value of `col5`; 9 bytes to store the value.
  Use the `disable varbinary truncation` configuration parameter to determine if trailing zeros are included at the end of `varbinary` and binary null data. This helps in situations where the exact number of trailing null bytes is significant to the value. See *Truncate Trailing Zeros* in the *Transact-SQL Users Guide* for more details.

Use the `decompression row threshold` configuration parameter to determine when the server uses row decompression instead of column by column decompression. If the table has more columns than the configuration value, row decompression is used instead of column decompression. For example, this changes the maximum number of columns in a table that remain uncompressed to 85:

```
sp_configure "decompression row threshold", 85
```

If the number of columns is greater than 85, SAP ASE uses row decompression.

2.2 Page-Level Compression

Use page-level compression to compress the amount of data redundancy on a page.

When you specify page-level compression for regular data, SAP ASE performs row-level compression first, then page-level compression.

Data pages often include repeated information (for example, the same date, time, or department ID). Instead of storing the same value multiple times, page-level compression lets you store these values in a single place and use a symbol on the data page to refer to them.

A number of techniques for page-level compression are used:

- Extracting repetitive information from variable-length byte strings and replacing them with shorter symbols.
  When you insert a new row into a data page, the data in the columns is compared with the symbols in the page dictionary. If a match is found in the dictionary for the new data, the dictionary symbol is stored instead of the data, and the row is compressed. When the data is retrieved, the symbol indicates the appropriate data. A page dictionary can include multiple entries, each with a different symbol that compresses a different piece of information.
- Extracting and removing short, duplicate values that use fixed-length columns from the rows.
  If a fixed-length column includes a high number of duplicates, SAP ASE stores the duplicate value in the page index, and uses a status bit in the row to indicate that this value is stored in the page index and is available for compression. When you retrieve data from the row, the status bit indicates the value that SAP ASE includes in the result set.
  A page index may contain multiple entries for different duplicate values in the page.
  For example, if you create this table:

  ```
  create table order_line (
     order_id int,
     disp_id tinyint,
     width_id smallint,
     number tinyint,
     info_id int,
     supply smallint,
     delivery datetime,
     quantity smallint,
  ```

```
   amount float,
   dist_info char(24))
lock datapages
```

And insert this data:

```
682, 1, 7, 11, 30000, 7, 'Dec 2 2008 1:19PM', 5, 290, 'Houston')
748, 1, 7, 12, 93193, 7, 'Sep 27 2009 1:15PM', 5, 9900, 'Bakersfield')
239, 1, 7, 13, 50383, 7, 'Aug 18 2008 11:47AM', 5, 8480, 'Modesto')
594, 1, 7, 14, 70901, 7, 'Aug 19 2008 10:37AM', 5, 84840, 'Houston')
849, 1, 7, 1, 3459, 7, 'July 10 2010 3:15PM', 5, 940, 'Alberta')
994, 1, 7, 2, 1232, 7, 'Jan 3 2010 2:15PM', 5, 848, 'Sonoma')
219, 1, 7, 3, 55341, 7, 'Feb 12 2008 9:26AM', 5, 4884, 'Vallejo')
004, 1, 7, 4, 98313, 7, 'Jan 19 2007 2:05PM', 5, 4484, 'Houston')
229, 1, 7, 5, 1347, 7, 'Aug 8 2009 3:37PM', 5, 448, 'Bakersfield')
394, 1, 7, 6, 51276, 7, 'Nov 10 2009 1:38PM', 5, 4473, 'Napa')
119, 1, 7, 1, 18089, 7, 'Oct 29 2009 12:56PM', 5, 312, 'Los Angeles')
938, 1, 7, 2, 38396, 7, 'June 1 2009 3:46PM', 5, 2248, 'Houston')
```

The `disp_id`, `width_id`, `supply`, and `quantity` columns all contain duplicate values (1, 7, 7, and 5), that are all short fixed-length columns, and candidates for page index compression.

- For `char` and `varchar` columns, frequently used characters are encoded with a representation that takes less storage.

If the row length after compression exceeds the original row length, the original row is used instead of the compressed row.

SAP ASE analyzes the data and automatically selects the appropriate method of page-level compression.

Compression does not automatically occur on a table configured for page-level compression until you insert a row that causes the page to become full.

3 Creating Databases for Data Compression

Compressed databases can include compressed and uncompressed tables or partitions.

Procedure

To create databases with data compression, use:

```
create database <database_name>
[…]
with dml_logging = { minimal | full }
, durability =
{ no_recovery | at_shutdown | full }
, compression = {none | row | page}
, index_compression  = { none | page }
```

The `compression =` parameter indicates that all tables in the database inherit the specified level of compression, unless you explicitly state otherwise, and `index_compression =` indicates you are creating a database that includes a compressed index. See the *Reference Manual: Commands > create database*.

> **i** Note
>
> The default setting for compression on the `model` database is `none`.

Example

This example creates the `emaildb` database with row-level compression on the `emaildb_dev` device:

```
create database emaildb
on emaildb_dev = '50M'
with compression = row
```

3.1 Altering the Compression Level of a Database

Changing a database's compression level does not change the compression level of existing tables in the database; only tables you create after you alter the database inherit the new compression level.

Procedure

Alter the compression level of existing databases using:

```
alter database <database_name>
 […]
set
    [[,] compression = {none | row | page}]
    [[,] index_compression = {none | page}]
```

See the *Reference Manual: Commands*.

Example

To alter the `pubs2` database to use page-level compression, use:

```
alter database pubs2
set compression = page
```

This alters the `pubs2` database to use page-level index compression:

```
alter database pubs2 set index_compression = page
```

4 Creating a Compressed Table

You can compress all tables except system and worktables.

Context

Use `create table` to create a compressed table, partition, or to designate index compression for a table. You need not compress all columns in a table. When designing your table, select the columns that offer the greatest benefit from compression. Partitions, and tables can use row- and page-level compression. Partitions for which you do not specify the compression level inherit the table-level compression.

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

Procedure

The partial syntax for compression is:

```
create table [<database>.[<owner>].]<table_name>
(<column_name> datatype ...
       [not compressed ],
 [, next_column...])
[with {max_rows_per_page = num_rows,
    ...
     compression [= {none | page | row }]
     index_compression [={none | page} ]
[on <segment_name>]
[partition clause]
<partition_clause>::=
partition by <partition_type> [(<column_name>[, <column_name>]...)]
 ([<partition_name>] ...
 [with compression = {none | page | row }] [on <segment_name>],
 [, <next_partition>...])
```

The `create table. . . with compression` parameter overrides the database-wide setting. That is, if you create a database with row-level compression, then issue a `create table` command that indicates page-level compression, the table is created using page-level compression.

Example

To compress all columns in the `sales` table, use:

```
create table sales
   (store_id int not null,
```

```
   order_num int not null,
   date datetime not null)
with compression = row
```

To compress only the `order_num` column, specify the other columns as `not compressed`:

```
create table sales
(  store_id int not null not compressed,
   order_num int not null,
   date datetime not null not compressed)
with compression = row
```

To use page-level compression on the `Y2008` partition and row-level compression on the `Y2009` partition, enter:

```
create table sales_date
    (store_id int not null,
     order_num int not null,
    date datetime not null)
partition by range (date)
(Y2008 values <= ('12/31/2008') with compression = page on seg1,
Y2009 values <= ('12/31/2009') with compression = row on seg2,
Y2010 values <= ('12/31/2010') on seg3)
```

Use `sp_help` to view a table's compression level. This is the `sp_help` compression information for the `mail` table:

```
 Name   Owner   Object_type   Object_status
        Create_date
 ----   -----   -----------   ------------------------------------------
        -------------------
 mail     dbo    user table   row level compressed, contains compressed data
        Apr  8 2011  2:55PM
```

4.1 Disabling Compression

Set the compression level to `none` to remove data compression from a table or partition.

Procedure

Disable compression using :

```
alter table <table_name>
set compression = none
```

> **i** Note
>
> Modifying a database's compression level, or enabling and disabling compression at table or partition level does not affect existing data; it affects only data you add or update after the change. However, changing whether a column is compressed or not performs a data copy, and therefore effects existing data.

See the *Reference Manual: Commands*.

Example

To set the compression to `none` for the sales table, use:

```
alter table sales
set compression = none
```

To disallow compression for the `order_num` column:

```
alter table sales
modify order_num int not compressed
```

To remove compression from the `Y2008` and `Y2009` partitions:

```
alter table sales_date
modify partition Y2008, Y2009 set compression = none
```

4.2 Altering the Compression Level of a Table or Partition

`alter table` does not affect the compression level of existing data, but affects the compression level of new or changed data rows produced by subsequent DML operations.

Context

`alter table` lets you:

- Enable compression on uncompressed tables or partitions, and disable compression on already compressed tables or partitions.
- Change the compression type (`row` or `page`) for compressed tables.
- Alter a column in a compressed table to allow or disallow compression.

> **i** Note
>
> You must set the compression level for a table before you can modify a column for compression.

Procedure

Alter the compression level of existing tables or partitions using:

```
alter table <table_name>
{
modify column [not] compressed
},
{
```

```
modify partition <partition_name>, [<partition_name> . . .]
set compression = {default | none | row | page}
},
{set compression = {none | page | row}}
```

See the *Reference Manual: Commands*.

Example

This example alters the `sales_data` table for compression:

```
alter table sales_data
set compression = row
```

This example modifies the `isbn` column for compression:

```
alter table sales_data
modify isbn compressed
```

5 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 database, table, index, and local index partition levels

Index compression is supported by:

- `reorg rebuild`
- DML
- triggers
- Data that has been bulk-copied into a table with indexes defined as compressed, is automatically compressed.
- Both `dbcc checktable` and `dbcc checkstorage` check the integrity of indexes
- `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.

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 database, table, index, or local index partition levels, use these commands:

- `alter database`
- `create database`
- `alter table`
- `create table`
- `alter index`
- `create index`
- `select into`

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

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

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.

5.1 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.

5.2 Creating an Index Compressed Table

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

The `create table with index_compression` command 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.

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.

For syntax, see the *Reference Manual: Commands*.

5.3 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.

The `create index with index_compression` command 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.

For syntax, see the *Reference Manual: Commands*.

5.4 Changing the Index 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.
- You must run `reorg rebuild` after changing the index compression state for the new compression state to take effect on existing index pages.

- 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.
- The permissions for `alter index` command 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.
- To remove compression from indexes, the indexes must be dropped and then re-created with `set index_compression off`.

For syntax, see the *Reference Manual: Commands*.

6 Datatypes Available for Compression

Not all datatypes are eligible for data compression.

Exact Numeric Integer Datatypes Eligible for Compression

Datatype	Length, in Bytes	Compression Type
`bigint`	8	Row and page dictionary
`int`	4	Row and page dictionary
`smallint`	2	Page index
`tinyint`	1	Page index
`ubigint`	8	Row and page dictionary
`unsigned int`	4	Row and page dictionary
`unsigned smallint`	2	Page index

- All exact numeric datatypes are compressed.
- Platform-specific big-endian and little-endian (most- and least-significant bytes) storage for exact numeric integers is in the specified number of bytes.

Exact Numeric Decimal Datatypes Eligible for Compression

Datatype	Length, in Bytes	Compression Type
`numeric (<precision>, <scale>)`	User-specified	Row and page dictionary
`decimal (<precision>, <scale>)`		

- All exact numeric decimal datatypes are compressed.
- Storage format for exact numeric decimals is a byte stream storing 1 byte for precision, 1 byte for scale, and `<n>` number of bytes for data.

Approximate Numeric Datatypes Eligible for Compression

Datatype	Length, in Bytes	Compressed?	Compression Type
`float (<precision>)`	4 bytes if precisions < 16, 8 if ≥ 16	No	N/A
`double precision`	8		
`real`	4		

Money Datatypes Eligible for Compression

Datatype	Length, in Bytes	Storage format	Compressed?	Compression Type
`money`	8	Two 4-byte values: one signed `int` and the other an `unsigned int`	Yes	Row and page dictionary
`smallmoney`	4	1 signed 4-byte integer		

Date and Time Datatypes Eligible for Compression

Datatype	Length, in Bytes	Storage Format	Compressed?	Compression Type
`bigdatetime`	8	Represented as an unsigned 64-bit integer. Using a base date of 1/1/0001, `bigdatetime` holds the number of microseconds between midnight of the base date and a point in time. Stores fractions of a second to 6 decimal places.	Yes	Row and page dictionary
`bigtime`	8	8-byte unsigned integer holding the number of microseconds since midnight. Stores fractions of a second to 6 decimal places.		

Datatype	Length, in Bytes	Storage Format	Compressed?	Compression Type
`date`	4	Stores the number of days, backward or forward, from January 1, 1900.		
`datetime`	8	Two 4-byte parts. First part stores the number of days forward or backward from 1/1/1900. Second part stores the number of 1/300th seconds since midnight.	Yes	Page dictionary (date portion) and row compressed (time portion)
`smalldatetime`	4	Two 2-byte unsigned `smallint` values. First stores the number of days since 1/1/1900. Second stores the number of minutes since midnight.	No	N/A
`time`	4	The number of milliseconds since midnight.	Yes	Row page dictionary

Character Datatypes Eligible for Compression

Datatype	Length, in Bytes	Storage Format	Compressed?	Compression Type
`char(<n>)`	User-specified	Single or multiple byte or character stream, depending on the character type	Yes	Row, page dictionary if length ≥ 4. Page index, if length < 4.
`unichar(<n>)`				
`nchar(<n>)`				
`varchar(<n>)`			Yes	Page dictionary if length ≥ 4. Page index, if length < 4.
`univarchar(<n>)`				
`nvarchar(<n>)`				

Binary Datatypes Eligible for Compression

Datatype	Length, in Bytes	Storage Format	Compressed?	Compression Type
`binary(<n>)`	User-specified	Byte stream	Yes (length ≥ 4)	Row, page dictionary if length ≥ 4. Page index, if length < 4.
`varbinary(<n>)`	User-specified		Yes (length ≥ 4)	Page dictionary if length ≥ 4. Page index, if length < 4.

Other Datatypes Eligible for Compression

Datatype	Length, in Bytes	Storage Format	Compressed?	Compression Type
`bit`			No	N/A
`timestamp`	8	Byte stream; binary data	No	N/A
`xtype_token`	User-specified		No	N/A
`text pointer`	16 bytes of binary data	Byte stream. 8 bytes of RID, 8 bytes of first text page's database timestamp value.	No	N/A

7 Compressed Data Storage Strategies

Pages in a compressed table may have a combination of row-compressed, page-compressed, and uncompressed data.

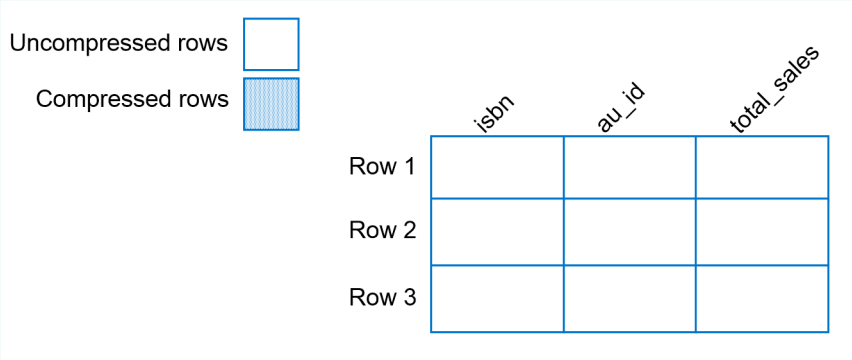
For example, if you create the `sales_data` table:

```
create table sales_data
(isbn bigint not null,
au_id varchar(11)not null,
total_sales int not null)
```

And insert this data:

```
4750984443, '903-94-9344', 34733
2385837442, '346-94-5593', 50945
2388347442, '346-94-5593', 50945
```

`sales_data` is uncompressed:



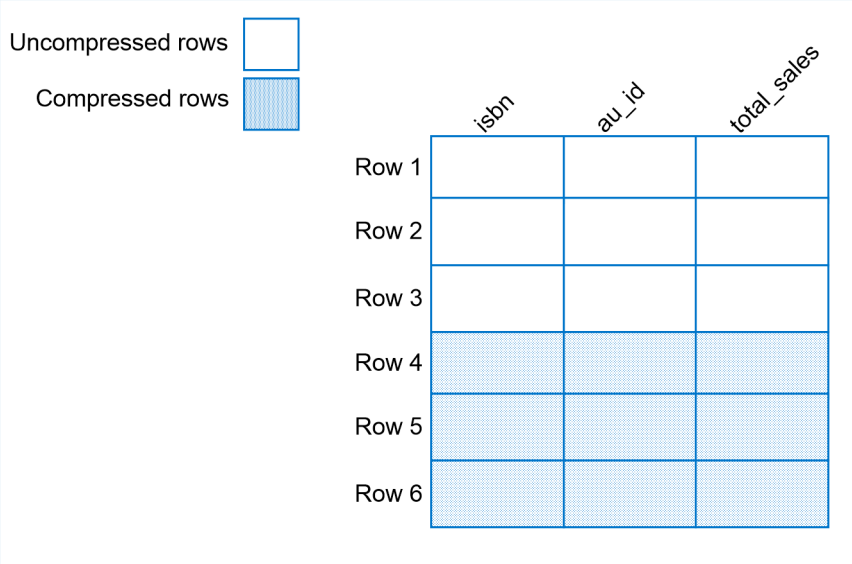
However, if you alter `sales_data` for compression:

```
alter table sales_data
set compression = row
```

And insert this data:

```
4783023685, '887-49-9984', 45009
3894350422, '776-45-9045', 89667
3349580094, '884-59-9983', 84855
```

Only the new data is compressed:



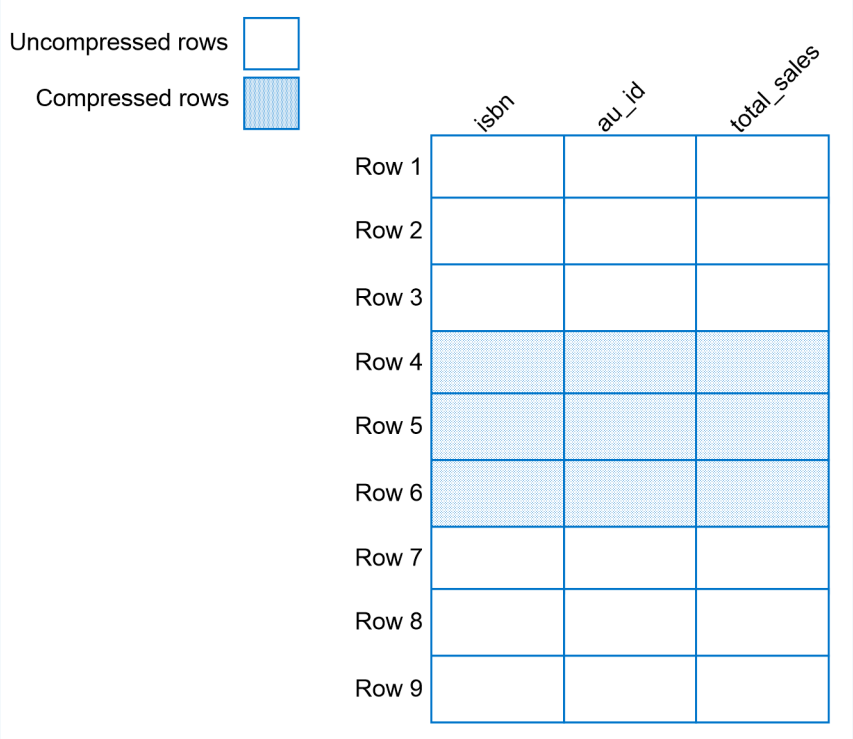
However, if you alter `sales_data` again to be uncompressed:

```
alter table sales_data set compression = none
```

And insert this data:

```
6590345093, '439-49-9943', 485844
3458940330, '559-40-3999', 21003
4859390403, '884-30-0200', 790499
```

SAP ASE does not compress the new data, but retains the older data in a compressed state:



`sp_help` reports whether a table has ever contained compressed data. This is the `sp_help` output for `sales_data`:

```
Name       Owner Object_type Object_status            Create_date
---------- ----- ----------- ------------------------ --------------------
sales_data dbo   user table  contains compressed data Apr  8 2011  4:36PM
```

8 Compressed Columns with Large Objects

SAP ASE lets you create databases and compress columns that use the `text`, `image`, `unitext`, in-row, and `java` large object (LOB) datatypes.

LOB columns can contain up to 2,147,483,647 (or 2^{31} -1) bytes of character or binary data. LOB values are stored on a text page chain. Only text pages are compressed.

In-row LOB compression is used if:

- The table is implicitly or explicitly row- or page-compressed, and,
- Any of the in-row large object columns in the table are implicitly or explicitly LOB compressed.

The table is implicitly or explicitly row- or page-compressed, and, Any of the in-row large object columns in the table are implicitly or explicitly LOB compressed.

SAP ASE uses the FastLZ (with LZO) and ZLib (with LZW.26) algorithm to compress LOB data. Both are dictionary-based compression techniques; that is, they replace repeated words on the data page with a status bit that points to the actual word in an index. The differences are:

- FastLZ – lower CPU usage and execution time.
- ZLib – higher compression ratio.

SAP ASE automatically determines the algorithm to use when you select the compression level. Levels 1 – 9 use the ZLib technique, and levels 100 and 101 use the FastLZ technique.

Generally, the higher the compression level, the more the LOB is compressed. However, the amount of compression depends on the content of the LOB. The higher the compression level, the more CPU-intensive the process, so a `<compression_level>` of 9 provides the best compression ratio, but also the heaviest CPU usage.

You can combine table-level and column-level compression.

Table 1: Combining Table- and Column-Level Compression

Compression Level	No Column Compression	Column is `not compressed`	Column uses `<compression_level>` Scale
No table-level compression	Uncompressed	Uncompressed	Column-level compression
`lob_compression` = 0	Uncompressed	Uncompressed	Column-level compression
`lob_compression` is the same as the table-level compression	Column level compression	Uncompressed	Column-level compression

The page layout is altered when LOB columns are compressed.

8.1 Creating a Compressed Database with LOB Datatypes

All tables in a database inherit the compression level you specify for LOB columns.

Procedure

1. Select a compression level to determine the database's compression strategy:

Option	Description
Compression Level	**Strategy**
1 – 9, where 9 provides the best compression ratio but heaviest CPU usage	Higher compression ratio (ZLib algorithm)
100 or 101	Lower CPU usage and execution time (FastLZ algorithm)

2. Create a database with LOB datatypes using

```
create database <database_name>
[…]
 with dml_logging = { minimal | full }
, durability =
{ no_recovery | at_shutdown | full }
, lob_compression = off | <compression_level>
```

The `lob_compression =` parameter indicates that all tables in the database inherit the specified level of compression for LOB columns.

Example

This creates the `email_lob_db`, which is configured for a LOB compression level of 101:

```
create database email_lob_db
on email_lob_dev = '50M'
with lob_compression = 101
```

8.2 Creating Compressed Tables with LOB Datatypes

You need not compress all columns in a table.

Procedure

1. Select a compression level to determine the table's compression strategy:

Option	Description
Compression level	**Strategy**
1 – 9, where 9 provides the best compression ratio but heaviest CPU usage	Higher compression ratio (ZLib algorithm)
100 or 101	Lower CPU usage and execution time (FastLZ algorithm)

2. Create a table with LOB compression using:

```
create table <table_name> (
<column_name> <data_type>
[compressed = <compression_level> | not compressed]
...
)
[with lob_compression = <compression_level>
```

The `compressed =` parameter controls column-level compression; `with lob_compression =` controls table-level compression.

Example

This example creates a compressed table that includes LOB data:

```
create table mail(user_name char(10),
mailtxt text compressed = 5,
photo image compressed = 1,
reply_mail text compressed = 9,
attachment image compressed = 100)
lock datarows
with lob_compression = 0
```

8.3 Altering Tables with Compressed LOB Datatypes

Use `alter table` command to enable or disable a table's compression.

Procedure

1. Select a compression level to determine the compression strategy for the table:

Option	Description
Compression Level	**Strategy**
1 – 9, where 9 provides the best compression ratio but heaviest CPU usage	Higher compression ratio (ZLib algorithm)
100 or 101	Lower CPU usage and execution time (FastLZ algorithm)

2. Alter a LOB table's compression level using:

```
alter table <table_name>
add <column_name> datatype ...
[compressed = <compression_level> | not compressed]
| set
  [, lob_compression = off | <compression_level> ]
| modify column_name ...
  [compressed = <compression_level> | not compressed ]
```

Example

This alters the compression level of the titles table to row:

```
alter table titles set compression = row
```

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

www.sap.com/contactsap

THE BEST RUN SAP