SYBASE®

An SAP® Company

New Features Guide

Adaptive Server® Enterprise

15.7 ESD #3

DOCUMENT ID: DC00641-01-1573-02
LAST REVISED: June 2013

Contents

sp_helpconfig

Adaptive Server® Enterprise version 15.7 ESD #3 adds the **estimate** parameter to **sp_helpconfig**, which determines the approximate amount of memory required for certain memory-intensive configuration parameters.

Currently, the **estimate** parameter works only with the **compression info pool size** configuration parameter.

Syntax

```
sp_helpcofig "config_name"
    [, { "size" | "estimate [using_argument = value [, using_argument
= value ] [, ...] ] } ]
```

Parameters

- **estimate –** recommends a value to which you can set the indicated configuration parameter, based on the settings of other configuration parameters or user-specified values that override those settings.
- ***using_argument* = *value* –** provides these additional arguments for the **estimate** parameter to override default values for configuration parameters that affect the estimated memory:
  - `maxconcusers = value` – specifies the maximum **number of concurrent users**, as an integer, that can access compressed tables.

    For example, `maxconcusers = 0.7` indicates 70 percent of the configured value for **number of user connections**. An integer value of 1 or greater specifies an absolute number of concurrent users.
  - `numcolumns = value` – specifies the average number of columns in a compressed table.
  - `numcompobjs = value` – specifies the default number of open objects as an integer, or as a percentage, that require memory for compression metadata. For example, `numcompobjs = 0.2` indicates that 20 percent of the configured value for **number of open objects**. An integer value of 1 or greater specifies an absolute number of open objects.
  - `numtables = value` – determines the average number of compressed tables accessed in a statement.

  Issuing **sp_helpconfig** without arguments generates usage information, showing the subclauses you may specify, and some examples of typical usage.

Examples

- **Example 1 –** shows the **sp_helpconfig ... estimate** parameter run from a system database (such as `master` or `tempdb`). In this example, **sp_helpconfig** performs the estimate using default values for factors that affect the required memory:

```
sp_helpconfig 'compression info pool', 'estimate'
The compression information pool size parameter indicates the
amount of memory currently available to store table compression
information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit              Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0     2147483647          4096           4096
           8240      memory pages(2k)  dynamic

Estimated memory required for 600 concurrent users requesting
memory from this pool, accessing 500 compressed objects with
50 columns, on an average, per compressed table is 22600 KB.

Configuration parameter, 'compression info pool size', can be
configured to 21971 to fit in 44200K of memory.
```

- **Example 2 –** overrides the defaults with site-specific parameters to estimate the memory and configuration value setting. **sp_helpconfig** is executed a second time from a system database (such as `master` or `tempdb`) to estimate the memory required for server-wide concurrent access to compressed objects, when these objects are accessed from multiple databases in the server:

```
sp_helpconfig 'compression info pool', 'estimate
 using numcompobjs=0.3, numtables=2.25, numcolumns=25,
 maxconcusers=0.85'

The compression information pool size parameter indicates the
amount of memory currently available to store table compression
information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit              Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0     2147483647           4096           4096
         8240        memory pages(2k)  dynamic

Estimated memory required for 1020 concurrent users requesting
memory from this pool, accessing 150 compressed objects with 25
columns, on an average,
 per compressed table is 37020 KB.

Configuration parameter, 'compression info pool size', can be
configured to 18402 to fit in 37020K of memory.
```

- **Example 3 –** shows **sp_helpconfig ... estimate** run against a user database with numerous compressed tables, which are used frequently by an application. The server is configured as:

```
sp_configure 'user connections', 900
sp_configure 'worker processes', 500
sp_configure 'max parallel degree', 5
```

  In this example, **estimate** gathers metrics from the user database from which you issue the procedure for:

  - The number of compressed objects
  - The average number of columns in these compressed objects

  Using these input values, **sp_helpconfig** estimates the memory required for `compression info pool` to store table compression information:

```
sp_helpconfig 'compression info pool size', 'estimate'

The compression information pool size parameter indicates
the amount of memory currently available to store table
compression information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit             Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0      2147483647          4096          15396
           33384     memory pages(2k)  dynamic

Estimated memory required for 1400 concurrent users requesting
memory from this pool, accessing 78240 compressed objects
with 10 columns, on an average, per compressed table is 74850 KB.

Configuration parameter, 'compression info pool size', can be
configured to 34519 to fit in 74850K of memory.
```

  This output indicates that a total of 1400 concurrent users are expected to simultaneously request memory. The database has slightly more than 78000 compressed objects, with each table having, on average, 10 columns. The estimated value for this configuration option is 34519.

  However, if not all the objects are routinely accessed simultaneously, and not all the configured user connections are simultaneously active, you can refine the estimates by providing site-specific overrides with the **using** parameter subclause:

```
sp_helpconfig 'compression info pool size', 'estimate
using numcompobjs = 50000, maxconcusers=600'

The compression information pool size parameter indicates
the amount of memory currently available to store table
compression information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit             Type
```

```
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
           0      2147483647          4096          15396
           33384     memory pages(2k)  dynamic

Estimated memory required for 1100 concurrent users requesting
memory from this pool, accessing 50000 compressed objects with
10 columns, on an average, per compressed table is 55225 KB.

Configuration parameter, 'compression info pool size', can be
configured to 25468 to fit in 55225K of memory.
```

In this output, `maxconcusers = 600` implies that 600 concurrent client connections are accessing compressed objects requesting memory. Because of the parallel configuration settings, **sp_helpconfig** estimates that a total of 1100 requesters may concurrently request memory. The estimated value for this configuration option is 25468.

monMemoryUsage

The `monMemoryUsage` monitoring table provides information about server and kernel memory pools, including metrics about their sizes, usage patterns, and availablility.

Columns

The columns for `monMemoryUsage` are:

Name	Datatype	Attribute	Description
`Flags`	`int`		Status flags that describe the memory pool.
`ConfigNum`	`int`		Configuration number for the primary configuration option controlling the size of the memory pool.
`TotalSize`	`bigint`		Total size, in bytes, of the memory pool.
`UsedSize`	`bigint`		Currently used size, in bytes, of the memory pool.
`FreeSize`	`bigint`		Amount of free memory, in bytes, in the pool.
`NumAllocs`	`bigint`	Counter	Total number of allocations requested.

Name	Datatype	Attribute	Description
`NumFrees`	`bigint`	Counter	Total number of frees performed. "Frees" are the opposite of allocations. Adaptive Server allocates memory fragments, and the number of allocations is tracked by `NumAllocs`. When the task is finished, the memory fragment is freed (that is, returned to the memory pool). `NumFrees` tracks the total number of these free operations
`NumSleeps`	`bigint`	Counter	Total number of sleeps encountered while allocating memory fragments.
`PoolOwnerKPID`	`int`		Kernel process ID (KPID) of task that owns this fragment of the memory pool.
`MemoryPoolName`	`varchar(32)`	Null, Parameter	Name of the memory pool.
`PoolType`	`varchar(30)`	Null, Parameter	Type of memory pool. One of: • Block • Object • Fragment • Stack
`ConfigOption`	`varchar(255)`	Null	Name of the primary configuration option controlling the size of the memory pool.
`NumSearches`	`bigint`	Counter, Null	Total number of free fragments examined before satisfying memory allocation requests from this memory pool.

Name	Datatype	Attribute	Description
NumRetries	bigint	Counter, Null	Number of retries performed for all free fragments.
ItemSize	int	Null	Size of an individual item (applies to object pool).
MinNumItems	int	Null	Minimum number of items in this pool.
MaxNumItems	int	Null	Maximum number of items in this pool.
NumUsedItems	int	Null	Number of used items in this pool.
NumItemsUsedHWM	int	Null	High-water mark for the number of items used in this pool.
MinUsedItemSize	bigint	Null	Size, in bytes, of smallest used item.
AvgUsedItemSize	bigint	Null	Average size, in bytes, of used items.
MaxUsedItemSize	bigint	Null	Size, in bytes, of largest used item.
NumUsedItemsMinSize	int	Null	Number of minimum-sized used items in this pool.
NumUsedItemsMaxSize	int	Null	Number of maximum-sized used items in this pool.
NumFreeItems		Null	Number of free items in this pool.
MinFreeItemSize	bigint	Null	Size, in bytes, of smallest item that is free.
AvgFreeItemSize	bigint	Null	Average size, in bytes, of free items.
MaxFreeItemSize	bigint	Null	Size, in bytes, of largest item that is free.
NumFreeItemsMinSize	int	Null	Number of minimum-sized free items in this pool.
NumFreeItemsMaxSize	int	Null	Number of maximum-sized free items in this pool.

Name	Datatype	Attribute	Description
`NumBlocks`	`int`	Null	Number of blocks of memory used for this pool.
`MemSize1`	`int`	Null	Memory pool specific request size 1, in bytes.
`NumUsedItemsSize1`	`int`	Null	Number of used items in this pool of size `MemSize1` .
`NumFreeItemsSize1`	`int`	Null	Number of free items in this pool of size `MemSize1` .
`MemSize2`	`int`	Null	Memory pool specific request size 2, in bytes.
`NumUsedItemsSize2`	`int`	Null	Number of used items in this pool of size `MemSize2`.
`NumFreeItemsSize2`	`int`	Null	Number of free items in this pool of size `MemSize2`.

Not all output from all `monMemoryUsage` columns applies, or is relevant to, all memory pools, and depending on the type of memory pool, you may need to select the relevant columns. Typically, columns return a value of NULL if they do not apply to a specific memory pool.

These columns report the metrics for used versus free fragments for memory pools of type `Fragment`:

- `NumUsedItems`
- `NumItemsUsedHWM`
- `MinUsedItemSize`
- `AvgUsedItemSize`
- `MaxUsedItemSize`
- `NumUsedItemsMinSize`
- `NumUsedItemsMaxSize`
- `NumFreeItems`
- `MinFreeItemSize`
- `AvgFreeItemSize`
- `MaxFreeItemSize`
- `NumFreeItemsMinSize`
- `NumFreeItemsMaxSize`

This example lists memory pools in the server, along with the primary configuration option affecting the size of the memory pool:

```
select PoolType = convert(varchar(10), PoolType),
MemoryPoolName = convert(varchar(30), MemoryPoolName),
ConfigOption = convert(varchar(30), ConfigOption)
from monMemoryUsage order by 1, 2
PoolType  MemoryPoolName          ConfigOption
--------  ----------------------- ----------------------------
Block     Compression             compression memory size
Block     Global Block Pool       NULL
Block     Kernel Resource Memory  kernel resource memory
Block     Proc Cache Header       procedure cache size
Block     Pss Heap Memory         heap memory per user
Block     RTMS Block Heap         messaging memory
Fragment  CPINFO memory pool      compression info pool size
Fragment  Column Default Pool     column default cache size
Fragment  Data Cache Frag         NULL
Fragment  Data Change Frag        NULL
Fragment  Data Transfer Utility   transfer utility memory size
[...]
```

This example lists the common metrics that are applicable to most memory pools:

```
select MemoryPoolName = convert(varchar(30), MemoryPoolName),
TotalSize, UsedSize, FreeSize, NumAllocs, NumFrees
from monMemoryUsage order by 1
```

This example lists the metrics that apply to fragment memory pools:

```
select MemoryPoolName = convert(varchar(30), MemoryPoolName),
NumUsedItems, NumItemsUsedHWM, MinUsedItemSize,
AvgUsedItemSize, MaxUsedItemSize
from monMemoryUsage
where PoolType = "Fragment"
MemoryPoolName  NumUsedItems  NumItemsUsedHWM  MinUsedItemSize
  AvgUsedItemSize  MaxUsedItemSize
--------------  ------------  ---------------  ---------------
  ---------------  ---------------
Pss Frag Pool         983048                0            16960
                0                0
Pss Frag Pool        1376267                0            17040
                0                0
Pss Frag Pool        2293778                0            16960
                0                0
```

spaceusage

Returns metrics for space use in Adaptive Server as a comma-separated string.

Syntax

```
spaceusage(database_id [, object_id [, index_id [,
partition_id ] ] ] )
```

Parameters

- ***database_id* –** a numeric expression that is the ID for a database. These are stored in the `dbid` column of `sysdatabases`.
- ***object_id* –** a numeric expression that is an object ID for a table object. These are stored in the `id` column of `sysobjects`.
- ***index_id* –** is the index ID of the object you are investigating. Depending on the *index_id* you use, **spaceusage** reports:
  - *index_id* = 0 – returns the space metrics for only the data layer of an object, including all its data partitions.
  - *index_id* = 1 – is applicable only for allpages-locked tables with a clustered index and returns the space metrics for only the index layer of the clustered index.
  - *index_id* > 1 – returns the space metrics for the index layer of the corresponding index.
  - *index_id* = 255 – returns the space metrics for off-row, large object page chains.
- ***partition_id* –** the ID of the partition for which space usage metrics are to be retrieved.

Examples

- **Example 1 –** returns space usage information for the entire database:

  ```
  select spaceusage()
  "reserved pages=1163, used pages=494, data pages=411, index
  pages=78,
  oam pages=83, allocation units=94, row count=50529, tables=33,
  LOB pages=3, syslog pages=8"
  ```

- **Example 2 –** returns space metrics for all the indexes on the object specified by **object_id**, including all partitions, if any, on each index, and the space used by off-row large object page chains:

  ```
  select spaceusage(dbid, objid)
  ```

- **Example 3 –** returns space metrics for the specified partition for the listed *object_id* and *index_id*:

  ```
  select spaceusage(database_id, object_id, index_id)
  ```

  The output from **spaceusage** run against a database containing numerous user objects is shown below. **spaceusage** reports the space metrics for the data layer and all the indexes on this table.

  ```
  select spaceusage(db_id(), object_id('syspartitions'))
  -----------------------------------------------------------------
  -----
  reserved pages=2220, used pages=2104, data pages=2100, index
  pages=1096, oam pages=4, allocation units=373, row count=174522,
  tables=1, LOB pages=0
  ```

  In this result, the reserved pages, used pages, and data pages values report the respective page counts for data and index pages. Because `index pages` reports the page counts for only the index pages of the three indexes on `syspartitions`, determine the number of data pages for only the data layer of this table by subtracting the value for index pages from the value for the data pages: 2100 - 1096 = 1004 pages.

Confirm the number of data pages for only the data layer of this table by executing **spaceusage** with a value for the *index_id* parameter of 0:

```
select spaceusage(db_id(), object_id('syspartitions'), 0)
-------------------------------------------------------------------
-----
reserved pages=1064, used pages=1005, data pages=1004, index
pages=0, oam pages=1, allocation units=229, row count=174522,
tables=1, LOB pages=0
```

spaceusage reports a value for data pages (1004), which is consistent with the equation above, and because the query requests space metrics for only the data layer, it returns a value of 0 for the index pages.

- **Example 4 –** returns the aggregate space metrics for all objects, including user and system catalogs, that occupy space in the database:

  ```
  select spaceusage(database_id)
  ```

  However, **spaceusage** does not report on tables that do not occupy space (for example, fake and proxy tables). Currently, **spaceusage** also does not report on `syslogs`.

Usage

Depending on which parameters you include, **spaceusage** may report on any or all of:

- `reserved pages` – number of pages reserved for an object, which may include index pages if you selected index IDs based on the input parameters.
- `used pages` – number of pages used by the object, which may include index pages if you selected index IDs based on the input parameters.

  The value for used pages that **spaceusage** returns when you specify *index_id* = 1 (that is, for all-pages clustered indexes) is the used page count for the index layer of the clustered index. However, the value the **used_pages** function returns when you specify *index_id* = 1 includes the used page counts for the data and the index layers.
- `data pages` – number of data pages used by the object, which may include index pages if you selected index IDs based on the input parameters.
- `index pages` – number of index-only pages, if the input parameters specified processing indexes on the objects. To determine the number of pages used for only the index-level pages, subtract the number of large object (LOB) pages from the number of index pages.
- `oam pages` – number of OAM pages for all OAM chains, as selected by the input parameters.

  For example, if you specify:

  ```
  spaceusage(database_id, object_id, index_id)
  ```

`oam pages` indicates the number of OAM pages found for this index and any of its local index partitions. If you run **spaceusage** against a specific object, `oam pages` returns the amount of overhead for the extra pages used for this object's space management.

When you execute **spaceusage** for an entire database, `oam pages` returns the total overhead for the number of OAM pages needed to track space across all objects, and their off-row LOB columns.

- `allocation units` – number of allocation units that hold one or more extents for the specified object, index, or partition. `allocation units` indicates how many allocation units (or pages) Adaptive Server must scan while accessing all the pages of that object, index, or partition.

  When you run **spaceusage** against the entire database, `allocation units` returns the total number of allocation units reserving space for an object. However, because Adaptive Server can share allocation units across objects, this field might show a number greater than the total number of allocation units in the entire database.
- `row count` – number of rows in the object or partition. **spaceusage** reports this row count as 0 when you specify the *index_id* parameter.
- `tables` – total number of tables processed when you execute **spaceusage** and include only the *database_id* parameter (that is, when you are investigating space metrics for the entire database).
- `LOB pages` – number of off-row large object pages for which the index ID is 255.

  `LOB pages` returns a nonzero value only when you use **spaceusage** to determine the space metrics for all indexes, or only the LOB index, on objects that contain off-row LOB data. `LOB pages` returns 0 when you use **spaceusage** to examine the space metrics only for tables (which have index IDs of 0).

  When you run **spaceusage** against the entire database, `LOB pages` displays the aggregate page counts for all LOB columns occupying off-row storage in all objects.

sybrestore

Use the **sybrestore** utility to restore an Adaptive Server database to the time of failure from the most current full database backup dump files.

sybrestore is supported on UNIX and Windows and supports Adaptive Server versions 15.7 and later.

sybrestore is installed as part of the Adaptive Server software. For information about how to install Adaptive Server, see the installation guide for your platform. The executable file is located in: `$SYBASE/ASE-15_0/bin/`

For information, see *Restoring a Database Using* ***sybrestore*** in the *Utility Guide*.

sp_chgattribute tabname

The **sp_chgattribute tabname** system procedure supports features introduced in Adaptive Server 15.7 SP100.

A new configuration value, 2, has been introduced for the **dealloc_first_txtpg** parameter. Setting the parameter to this value allows you to not deallocate the first text page after NULL update, as a result, the text pointer will not be set to NULL even after null update.

Whether the first text page will be deallocated after NULL update depends on the combination of this table parameter and the database option **deallocate first text page**.

```
DB setting (deallocate first text page)    |   0   1   2
---------------------------------------------------------
dealloc_first_txtp - true                  |   Y   Y   N
dealloc_first_txtp - false                 |   N   N   N
```

- Y: deallocate first text page after null update
- N: not deallocate first text page after null update

The output from **sp_help** indicates whether first text page will be deallocated, as seen below.

```
> sp_chgattribute  mytab, "dealloc_first_txtpg", 1
'dealloc_first_txtpg' attribute of object 'mytab' changed to 1.
(return status = 0)
1>
2> sp_help mytab
  Name   Owner   Object_type       Object_status             Create_date
------- ------- ------------- -------------------------- --------------------
 mytab    dbo    user table   deallocate first text page  Jan 22 2013  9:45PM

> sp_chgattribute  mytab, "dealloc_first_txtpg", 2
'dealloc_first_txtpg' attribute of object 'mytab' changed to 2.
(return status = 0)
1>
2> sp_help mytab
  Name   Owner   Object_type       Object_status          Create_date
------- ------- ------------- ---------------------- --------------------
 mytab      dbo     user table    keep first text page   Jan 22 2013  9:45PM
```

- Y: deallocate first text page after null update
- N: not deallocate first text page after null update

To set the parameter, use:

```
sp_chgattribute tabname, "dealloc_first_txtpg", 0 | 1 | 2
```

- 0 – default, existing value, if either the table option setting is 1, or the database option **deallocate first text page** is TRUE, then deallocate the first text page after NULL update; otherwise, do not deallocate the first text page.

- 1 – deallocate the first text page after NULL update (overriding the setting of the database option **deallocate first text page**).
- 2 – do not deallocate the first text page after NULL update (overriding the setting of the database option **deallocate first text page**).

writetext

The **writetext** command, which permits minimally logged, interactive updating of an existing `text`, `unitext`, or `image` column, includes changes for Adaptive Server 15.7 ESD #3.

writetext
writetext includes these changes:

- **writetext** logs operations on in-row LOB columns.
- You can run the **writetext** command (with or without the **with log** parameter) simultaneously with the **online** parameter.

allocinfo

Returns a list of allocation pages that are stored in an object allocation map (OAM) page.

Syntax

```
allocinfo(dbid, pageid, "option")
```

Parameters

- ***dbid* –** is the database ID.
- ***pageid* –** is the page ID.
- ***option* –**
  - **help** – shows a message with the different options.
  - **alloc pages on oam** – provides the allocation page information.

Examples

- **–** provides a list of allocation pages that are stored in an object allocation map (OAM) page.

  ```
  select allocinfo(1,888,"alloc pages on oam")
  ```

Usage

allocinfo

- Mechanism to retrieve all allocation pages for a particular partition or index.
- Returns NULL for an invalid page, on a page other than OAM page, *option* values

Permissions

You must have sa_role to execute this command.

sp_dboption

Adaptive Server 15.7 ESD #3 introduces a new parameter for the **sp_dboption** system procedure.

deallocate first text page

The values for **deallocate first text page** are:

- false – default, do not deallocate the first text page after a NULL update.
- true – deallocate the first text page after a NULL update.

For example:

```
use master
go
sp_dboption mydb, "deallocate first text page", true
go
```

Index