Day2 - SAP Sybase ASE Session

SAP Sybase ASE 15.7

Training Agenda

	Sr No	Topic details
Day 1 and 2	1	Introduction to ASE
	2	ASE server components
	3	ASE multiple databases
	4	System tables & procedures
	5	ASE directory structure
	6	Starting & stopping server
	7	ASE configuration file
		Lab exercise
Day 3 and 4	8	ASE memory components
	9	Cache types and usage
	10	ASE devices
	11	Database and options
	12	Manage database in ASE
	13	Administering system roles and logins
	14	Managing database access and users
	15	Implementing object permissions, groups and roles
	16	Database Backup & recovery
		Lab exercise
Day 5 and 6	17	Sybase utilities
	18	Monitoring ASE using dbacockpit
	19	Maintenance tasks and commands
	20	Basic understanding of sp_sysmon
	21	Sybase SP patching
		Lab exercise

How SAP ASE Allocates Memory

The size of SAP ASE logical pages (2, 4, 8, or 16K) determines the server's space allocation.

Each allocation page, object allocation map (OAM) page, data page, index page, text page, and so on are built on a logical page. For example, if the logical page size of SAP ASE is 8K, each of these page types are 8K in size. All of these pages consume the entire size specified by the size of the logical page. Larger logical pages allow you to create larger rows, which can improve your performance because SAP ASE accesses more data each time it reads a page. For example, a 16K page can hold 8 times the amount of data as a 2K page, an 8K page holds 4 times as much data as a 2K page, and so on, for all the sizes for logical pages.

The logical page size is a server-wide setting; you cannot have databases that have various sizes of logical pages within the same server. All tables are appropriately sized so that the row size is no greater than the current page size of the server. That is, rows cannot span multiple pages.

Minimum database sizes

Logical page size	Minimum database size
2K	2MB
4K	4MB
8K	8MB
16K	16MB

Disk Space Allocation

The logical page size is not the same as the memory allocation page size. Memory allocation page size is always 2K, regardless of logical page size, which can be 2, 4, 8, or 16K.

Most memory-related configuration parameters use units of 2K for their memory page size, including:

- **max memory**
- **total logical memory**
- **total physical memory**
- **procedure cache size**
- **size of process object heap**
- **size of shared class heap**
- **size of global fixed heap**

CONFIGURATION

Dynamic Configuration
Most of the configuration parameters are dynamic; there is no need to reboot the ASE server for changes to take effect. The dynamic configuration allows easy reconfiguration, even in production environments.

Configuration of Physical Memory
The total physical memory that ASE uses is limited by the *max memory* configuration parameter. This memory is assigned for different use cases inside the DBMS. In SAP ASE, the most important memory pools are:

- Caches for storing data and index pages
- Table, index and partition metadata caches
- Procedure cache, which is used to compile, execute, and cache query access plans
- Lock list used for row and table locks
- Memory required for user connections

Number of CPU Cores
The number of CPU cores that SAP ASE is allowed to use can be configured by the maximum number of ASE engines and the number of threads in the ASE thread pools.

Memory Management in SAP ASE

Memory exists in SAP ASE as total logical or physical memory.

- Total logical memory – is the sum of the memory required for all the **sp_configure** parameters. The total logical memory must remain available, but may or may not be in use at a given moment. The total logical memory value may change due to changes in the configuration parameter values.
- Total physical memory – is the sum of all shared memory segments in SAP ASE. That is, total physical memory is the amount of memory SAP ASE uses at a given moment. You can verify this value with the read-only configuration parameter **total physical memory**. The value of **total physical memory** can only increase because SAP ASE does not shrink memory pools once they are allocated. You can decrease the amount of total physical memory by changing the configuration parameters and restarting SAP ASE.

When SAP ASE starts, it allocates:

- Memory used by SAP ASE for nonconfigurable data structures
- Memory for all user-configurable parameters, including the data cache, the procedure cache, kernel resource memory, and the default data cache.

Determine the SAP ASE Memory Configuration

The total memory allocated during system start-up is the sum of memory required for all the configuration needs of SAP ASE. You can obtain this value from the read-only configuration parameter **total logical memory**.

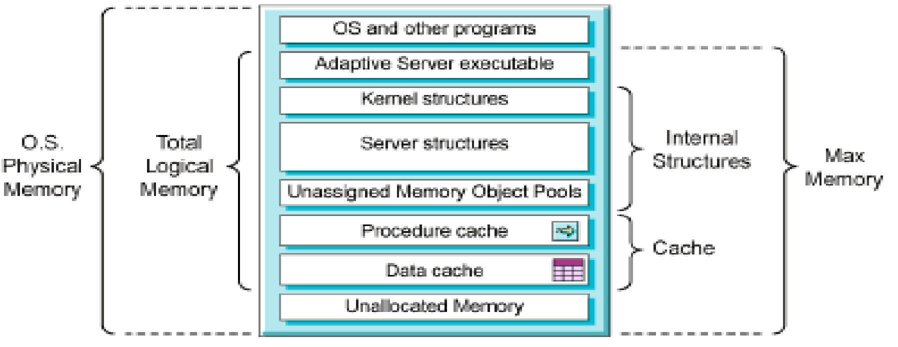
This value is calculated by SAP ASE. The configuration parameter **max memory** must be greater than or equal to **total logical memory**. **max memory** indicates the amount of memory you will allow for SAP ASE needs.

During server start-up, by default, SAP ASE allocates memory based on the value of **total logical memory**. However, if the configuration parameter **allocate max shared memory** has been set, then the memory allocated will be based on the value of **max memory**. The configuration parameter **allocate max shared memory** enables a system administrator to allocate the maximum memory that is allowed to be used by SAP ASE, during server start-up.

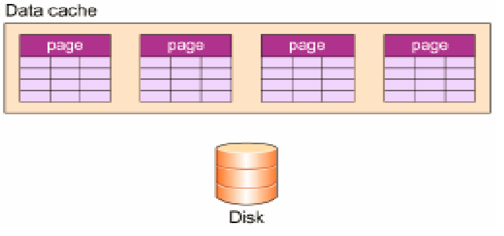
For example, if you set **allocate max shared memory** to 0 (the default) and **max memory** to 500MB, but the server configuration requires only 100MB of memory at start-up, SAP ASE allocates the remaining 400MB only when it requires the additional memory. However, if you set **allocate max shared memory** to 1, SAP ASE allocates the entire 500MB when it starts.

If **allocate max shared memory** is set to 0 and you increase **max memory**, the actual memory allocation happens when it is needed. If **allocate max shared memory** is set to 1and you increase **max memory**, SAP ASE attempts to allocate memory immediately. If the allocation fails, SAP ASE writes messages to the error log.

Adaptive Server Memory Components

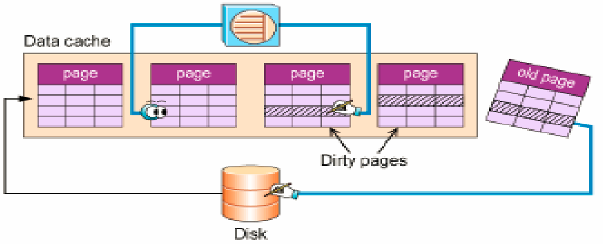


Data Cache



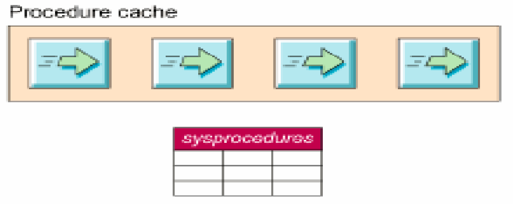
- Data cache is a portion of Adaptive Server memory where data, index, and log pages currently in use by the server are held
- When pages are not in use by the server, they are stored only on disk

How a Page Cycles Through Data Cache



- When a data modification statement is executed:
  1. Adaptive Server copies the related page into data cache.
     - If the page is already in data cache, this step is skipped.
  2. The page in data cache is read and modified as needed.
  3. Eventually, the page is "aged out."
     - This occurs when Adaptive Server has read enough new pages into data cache so there is no longer room for the least recently used pages.
     - Any changes made to a page while it was in data cache are written to disk before the page is aged out.

Procedure Cache



- Procedure cache is a portion of Adaptive Server memory where query plans currently in use by the server are held

Caches and Performance

- The data and procedure caches are designed to improve server performance
  - Frequently used pages and query plans remain in cache
    - There is less I/O from reading and writing pages to disk
    - New query plans do not need to get generated as often
- If either cache is too small, performance can suffer
  - Frequently used pages and query plans get aged out more quickly
    - More I/O occurs to read and write pages to disk
    - New query plans are generated more frequently

Memory Allocation at Startup

- During startup, Adaptive Server does the following:
  1. Verifies that there is memory available to support the current configuration.
     - If there is not sufficient memory available the server cannot start
  2. Allocates memory for the server executable
  3. Allocates memory for the kernel structures
  4. Allocates memory for the server structures specified in the configuration file
  5. Allocates memory for the procedure and data caches.

Data Cache Memory Allocation

Example:

```
sp_cacheconfig
 Cache Name           Status Type     Config Value Run Value
 -------------------- ------ -------- ------------ ------------
 default data cache   Active Default      0.00 Mb      8.00 Mb
                                  ------------ ------------
                          Total       0.00 Mb      8.00 Mb
==========================================================================
Cache: default data cache, Status: Active, Type: Default
 Config Size: 0.00 Mb,    Run Size: 8.00 Mb
 Config Replacement: strict LRU, Run Replacement: strict LRU
 Config Partition:  1,    Run Partition:  1
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb   1638 Kb      0.00 Mb      8.00 Mb     10
```

Major Uses of Adaptive Server Memory

- The aspects of Adaptive Server that typically use the most memory are:
  - Adaptive Server executable code and overhead
    - The System Administrator has no control over this
  - User connections
  - Open databases, open indexes, and open objects
  - Number of locks
  - Database devices
  - Data and procedure cache

Estimating Configuration Parameter Size

Syntax:

```
sp_helpconfig "parameter_name" , "size"
```

Examples:

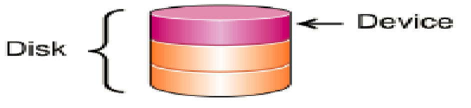
```
sp_helpconfig "number of user connections", "100"

...
Configuration parameter, 'number of user
connections', will consume 8998K of memory if
configured at 100....

sp_helpconfig "number of remote connections", "50M"

...
Configuration parameter, 'number of remote
connections', can be configured to 31030 to fit in
50M of memory.
```

sp_monitorconfig

Example:

```
sp_monitorconfig "number of open objects"

Usage information at date and time: Aug 16 2005
  1:29PM.
 Name                                  Num_free  Num_active
 Pct_act   Max_Used   Num_Reuse
 ------------------------- -------- ----------
 -------  --------   ---------
 number of open objects            358         142
   28.40        144             0
(return status = 0)
```

Devices



- A database device is a physical resource that stores the objects that make up the database
  - The term "device" does not necessarily refer to a distinct physical device
    - It can be any piece of disk, such as a disk partition
    - It can be a file in the operating system

Initializing Devices

- Device initialization is a process that prepares the device for storage and makes it known to the server
  - Devices must be initialized before they can be used
- Once a device has been initialized, it can be used to store:
  - Databases or specific database objects
  - Database transaction logs
- Devices are initialized using the **disk init** command
  - Maps the specified physical disk device or operating system file to a database device name
  - Lists the new device in master..sysdevices
  - Prepares the device for database storage
  - Only System Administrators can execute disk init

disk init Examples

UNIX/Linux example:

- Raw Partition

```
disk init name = "dev_dat_2",
physname = "/dev/rxyld",
vdevno 8, size = 5120
```

- File System

```
disk init  name = "dev_dat_2"
physname = "/betadisk/devices/dev_dat_2.dat",
vdevno 8, size = 5120
```

Windows example:

```
disk init  name = "dev_dat_2",
physname = "d:\devices\userdisk.dat",
vdevno 8, size = 5120
```

Viewing Device Information

Syntax: `sp_helpdevice [logical_device_name]`

- With a device name, it returns information about that device
- Without one, it returns information about all devices

Example: `sp_helpdevice dev_dat_2`

```
device_name          physical_name
description
status   cntrltype   vdevno      vpn_low     vpn_high
---------------------------------------------------
dev_dat_2            c:\devices\userdisk.dat
special,dsync on,directio off,physical disk,10.00 MB
16386           0        3            0         5119
```

6 - 19

sysdevices

- The master database system table that records each device

Example: (selecting a few columns from the table)

```
select low, high, status, cntrltype, name,
phyname from sysdevices

low high   status cntrltype name          phyname
--- ----- ------ --------- ---------------------------
 0  25599      3         0 master         /master.dat
 0  67583  16386         0 sysprocsdev …/sybprocs.dat
 0   2559  16386         0 systemdbdev  /sybdb.dat
 0  20000     16         2 tapedump1     /dev/nst0
 0  20000     16         3 tapedump2     /dev/nst1
```

disk resize

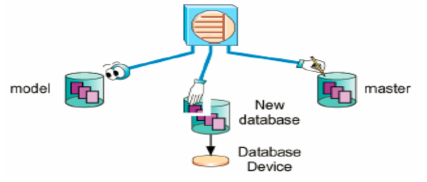
Syntax:

```
disk resize
name = <device_name>,
size = <additional_size>
```

- The device is online and available to users during resizing operation
- Mirroring (covered later in this module) must be disabled during resizing operation
- After resizing a device, execute **alter | create database** to utilize the additional space

6 - 22

What Happens When You Create a Database?

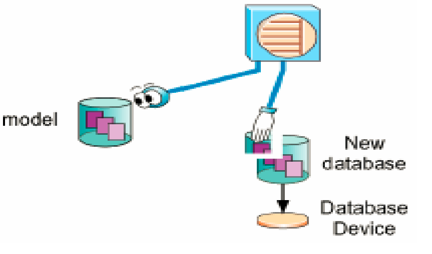


7 - 5

1. When a database is created:
2. The server reserves space on the specified device(s) for the data and the transaction log.
   - If no device is specified, space is used from the pool of default devices.
3. The server copies all objects in the model database into the new database.
4. The remaining pages are then formatted
5. The database options associated with model are applied to the new database.
6. Entries for the new database are inserted into the following tables in master:
   - sysdatabases.
   - sysusages.

Database Creation and *model*

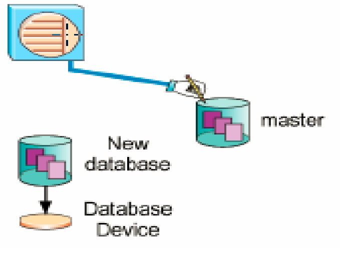
- model acts as a "template database"
  - The contents of model are always copied to the new database
  - You can create stored procedures, tables, rules, user-defined datatypes, users, privileges, and options in model
  - All future databases automatically inherit these objects and options



7 - 6

Database Creation and master

- System tables in master affected by database creation:
  - sysdatabases
    - Contains a row for every database on Adaptive Server, which specifies the database name and owner
    - Specifies a database ID, dbid, for each database
  - sysusages
    - Contains a row for every database fragment, indicating the size and logical starting disk address for that fragment



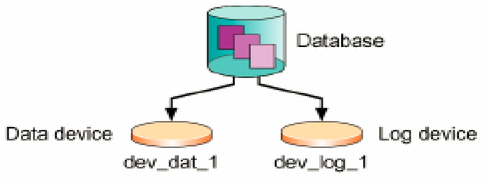
7 - 7

Sizing the Log

- Log size depends on:
  - The type and quantity of transactions
  - The frequency of log backups
- A good starting value is 10% to 25% of the overall database size
  - All inserts, deletes, and updates are logged
  - For create index, writetext, truncate table, select into and fast bulk copy, only space allocation and deallocation is logged
- Like databases, the log is easy to expand, but it cannot directly be shrunk

Data and Logs on Separate Physical Devices

- For each database, data and the log should be placed on separate devices
  - Allows transaction log backups to be performed
  - Lets you establish a fixed log size to keep it from competing for space with other database activity
  - Improves performance
  - Decreases the likelihood that both the database and the log will be damaged at the same time



Creating Databases: Examples

```
create database pubs2
```

- Both data and log portions are on a single default device
- The size of model or the *default database size*, whichever is larger

```
create database employeedb
on dev_dat_1 = '512000K'
```

- Both data and log portions are on the dev_dat_1 device
- The size is 500MB (500 * 1024K)

```
create database salesdb
on dev_dat_1 = 500
log on dev_log_1 = 200
```

- The data is on *dev_dat_1*, and its size is 500MB
- The log is on *dev_log_1*, and its size is 200MB
- The total size of the database is 700MB

Dropping Databases

Syntax: `drop database database_name`

Example: `drop database pubs2`

- Can be executed only by:
  - Owner of the database
  - System Administrator
- The database cannot be in use
- You should drop a database:
  - To remove experimental or old databases, thereby reclaiming space
  - Prior to recovering a database that has been damaged

Displaying Database Information

Syntax: `sp_helpdb [db_name]`

Example: `sp_helpdb salesdb`

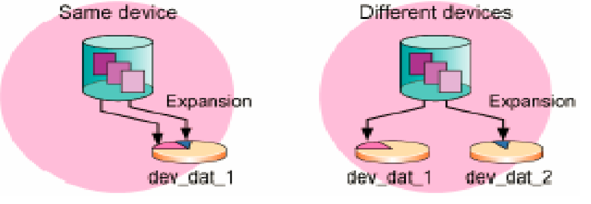
```
name     db_size      owner dbid ...
----     --------     ----- -----...
salesdb 700.0 MB         sa      6...

device_fragments size      usage...
---------------- ----      -----...
dev_dat_1        500.0 MB data only
dev_log_1        200.0 MB log only
```

Databases Run Out of Space

- The data portion and the log of a given database fill up as users use the database
  - When this happens, the default behavior is for all data modifications to be suspended
- If the data portion runs out of space:
  - Attempt to reclaim space by archiving old data
  - Expand the data portion
- If the log runs out of space:
  - Dump and truncate the log
  - Expand the log

Expanding a Database

- Database Owners and System Administrators can allocate additional space to a database
  - It can be on the same device or on a different device

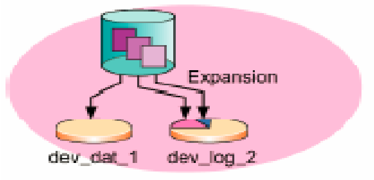


- Databases are expanded via the alter database command

Expanding the Log: Examples

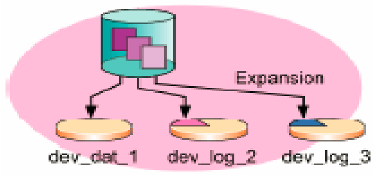
- On the same device:

```
alter database pubs2
log on dev_log_2 = 100
```



- Onto a new device:

```
alter database pubs2
log on dev_log_3 = 40
```



Moving the Log to a New Device

- For a database created with data and log on the same device, you can separate and move the log to a new device:
  - Build a new device using disk init
  - Alter the database onto the new device
  - Clean out old transaction log records using
    - dump tran with truncate_only
  - Execute sp_logdevice to make the newly created device a log only device
    - `sp_logdevice db_name, device_name`
  - Dump the database and master when script is complete.
- Effect
  - No new log pages are allocated on the old devices
  - Existing records deallocated as the log gets truncated
  - Log grows onto the new device
- Perform these steps with minimal users in the system and minimal time elapsed between steps

- The following commands can be used to monitor space usage:
  - **sp_helpdb**
  - **sp_helpsegment**
  - **sp_spaceused**

sp_helpdb

Syntax: `sp_helpdb [database_name]`

Example: `sp_helpdb sales`

```
-
name db_size  owner dbid  created       status
----- ------- ----- ----- -----------   ----------
sales 4 MB      sa    5     Oct 16 1992   no options
                                           set
device_fragments   size    usage ... free kbytes
----------------- ------ ----- ------------
dev_dat_1          2 MB    data only ... 1376
dev_log_2          1 MB    log only ... not applicable
dev_dat_3          1 MB    data only ... 1008
---------------------------------
log only free kbytes: 1008
  ...
```

sp_spaceused

- Displays free space within tables or in the database

Syntax:

```
sp_spaceused [object_name]
```

Examples:

```
sp_spaceused titles
name    rowtotal reserved data index_size unused
----    -------- -------- ---- ---------- ------
titles     18      48 KB   6 KB    4 KB     38 KB

sp_spaceused
db_name db_size reserved data index_size unused
------- ------  -------- ---- ---------- ------
pubs2    2.0 MB   1386 KB  452   94 KB     840 KB
```

8 - 23

- **reserved** - Space reserved for objects that have been defined.
- **data** - Space containing actual data.
- **index_size** - Space taken up by index.
- **unused** - Space reserved that does not yet have data.

Locking mechanism in ASE

Overview of locking

Consistency of data means that if multiple users repeatedly execute a series of transactions, the results are correct for each transaction, each time. Simultaneous retrievals and modifications of data do not interfere with each other: the results of queries are consistent.

For example, in Table 2-1, transactions T1 and T2 are attempting to access data at approximately the same time. T1 is updating values in a column, while T2 needs to report the sum of the values.

Table 2-1: Consistency levels in transactions

T1	Event Sequence	T2
`begin transaction`	T1 and T2 start.	`begin transaction`
`update account set balance = balance - 100 where acct_number = 25`	T1 updates balance for one account by subtracting $100.	
	T2 queries the sum balance, which is off by $100 at this point in time—should it return results now, or wait until T1 ends?	`select sum(balance) from account where acct_number < 50` `commit transaction`
`update account set balance = balance + 100 where acct_number = 45` `commit transaction`	T1 updates balance of the other account by adding the $100.	
	T1 ends.	

If transaction T2 runs before T1 starts or after T1 completes, either execution of T2 returns the correct value. But if T2 runs in the middle of transaction T1 (after the first update), the result for transaction T2 will be different by $100. While such behavior may be acceptable in certain limited situations, most database transactions need to return correct consistent results.

By default, Adaptive Server locks the data used in T1 until the transaction is finished. Only then does it allow T2 to complete its query. T2 "sleeps," or pauses in execution, until the lock it needs it is released when T1 is completed.

The alternative, returning data from uncommitted transactions, is known as a **dirty read**. If the results of T2 do not need to be exact, it can read the uncommitted changes from T1, and return results immediately, without waiting for the lock to be released.

Locking is handled automatically by Adaptive Server, with options that can be set at the session and query level by the user. You must know how and when to use transactions to preserve the consistency of your data, while maintaining high performance and throughput.

ASE Locking schemes

Allpages locking

Allpages locking locks both data pages and index pages. When a query updates a value in a row in an allpages-locked table, the data page is locked with an exclusive lock. Any index pages affected by the update are also locked with exclusive locks. These locks are transactional, meaning that they are held until the end of the transaction.

Datapages locking

In datapages locking, entire data pages are still locked, but index pages are not locked. When a row needs to be changed on a data page, that page is locked, and the lock is held until the end of the transaction. The updates to the index pages are performed using latches, which are non transactional. Latches are held only as long as required to perform the physical changes to the page and are then released immediately.

Datarows locking

In datarows locking, row-level locks are acquired on individual rows on data pages. Index rows and pages are not locked. When a row needs to be changed on a data page, a non transactional latch is acquired on the page. The latch is held while the physical change is made to the data page, and then the latch is released. The lock on the data row is held until the end of the transaction. The index rows are updated, using latches on the index page, but are not locked.

For each locking scheme, Adaptive Server can choose to lock the entire table for queries that acquire many page or row locks, or can lock only the affected pages or rows.

Server-wide locking scheme can be set using System Procedure – “sp_configure”
Ex: **sp_configure "lock scheme", 0, datapages (**This command sets the default lock scheme for the server to datapages**)**
When you first install Adaptive Server, lock scheme is set to ‘Allpages’.

Types of Locks in ASE

Shared locks
Adaptive Server applies shared locks for read operations. If a shared lock has been applied to a data page or data row or to an index page, other transactions can also acquire a shared lock, even when the first transaction is active. However, no transaction can acquire an exclusive lock on the page or row until all shared locks on the page or row are released. This means that many transactions can simultaneously read the page or row, but no transaction can change data on the page or row while a shared lock exists. Transactions that need an exclusive lock wait or "block" for the release of the shared locks before continuing.

Exclusive locks
Adaptive Server applies an exclusive lock for a data modification operation. When a transaction gets an exclusive lock, other transactions cannot acquire a lock of any kind on the page or row until the exclusive lock is released at the end of its transaction. The other transactions wait or "block" until the exclusive lock is released.

Update locks
Adaptive Server applies an update lock during the initial phase of an update, delete, or fetch (for cursors declared for update) operation while the page or row is being read. The update lock allows shared locks on the page or row, but does not allow other update or exclusive locks. Update locks help avoid deadlocks and lock contention. If the page or row needs to be changed, the update lock is promoted to an exclusive lock as soon as no other shared locks exist on the page or row

In general, read operations acquire shared locks, and write operations acquire exclusive locks.

Locking and performance

Locking affects performance of Adaptive Server by limiting concurrency. An increase in the number of simultaneous users of a server may increase lock contention, which decreases performance. Locks affect performance when:

•Processes wait for locks to be released – Any time a process waits for another process to complete its transaction and release its locks, the overall response time and throughput is affected.

•Transactions result in frequent deadlocks – A deadlock causes one transaction to be aborted, and the transaction must be restarted by the application. If deadlocks occur often, it severely affects the throughput of applications. Using datapages or datarows locking, or redesigning the way transactions access the data can help reduce deadlock frequency.

•Creating indexes locks tables – Creating a clustered index locks all users out of the table until the index is created; Creating a non-clustered index locks out all updates until it is created. Either way, you should create indexes when there is little activity on your server.

Reducing lock contention

Lock contention can impact Adaptive Server's throughput and response time. You need to consider locking during database design, and monitor locking during application design. Solutions include changing the locking scheme for tables with high contention, or redesigning the application or tables that have the highest lock contention.

For example:

- Add indexes to reduce contention, especially for deletes and updates.
- Keep transactions short to reduce the time that locks are held.

Locking tools

sp_who, sp_lock report on locks held by users, and show processes that are blocked by other transactions. Getting information about blocked processes sp_who reports on system processes. If a user's command is being blocked by locks held by another task or worker process, the status column shows "lock sleep" to indicate that this task or worker process is waiting for an existing lock to be released. The blk_spid or block_xloid column shows the process ID of the task or transaction holding the lock or locks.

You can add a user name parameter to get sp_who information about a particular Adaptive Server user. If you do not provide a user name, sp_who reports on all processes in Adaptive Server.

Viewing locks

To get a report on the locks currently being held on Adaptive Server, use sp_lock:

```
                  sp_lock
fid spid loid locktype         table_id   page  row dbname    context
--- ---- ---- ---------------- ---------- ----- --- --------  ----------------
  0  15   30  Ex_intent         208003772     0   0 sales     Fam dur
  0  15   30  Ex_page           208003772  2400   0 sales     Fam dur, Ind pg
  0  15   30  Ex_page           208003772  2404   0 sales     Fam dur, Ind pg
  0  15   30  Ex_page-blk       208003772   946   0 sales     Fam dur
  0  30   60  Ex_intent         208003772     0   0 sales     Fam dur
  0  30   60  Ex_page           208003772   997   0 sales     Fam dur
```

How to identify Concurrency problem?

Identifying tables where concurrency is a problem

sp_object_stats prints table-level information about lock contention. You can use it to:

- Report on all tables that have the highest contention level
- Report contention on tables in a single database
- Report contention on individual tables

The syntax is:

sp_object_stats *interval* [, *top_n* [, *dbname* [, *objname* [, *rpt_option*]]]]

To measure lock contention on all tables in all databases, specify only the interval. This example monitors lock contention for 20 minutes, and reports statistics on the ten tables with the highest levels of contention:

```
sp_object_stats "00:20:00"
```

Additional arguments to sp_object_stats are as follows:

- *top_n* – allows you to specify the number of tables to be included in the report. Remember, the default is 10. To report on the top 20 high-contention tables, for example, use:

  ```
  sp_object_stats "00:20:00", 20
  ```

How to identify concurrency problem? Contd…

Here is sample output for titles, which uses datapages locking:

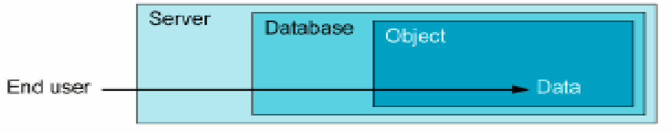
```
Object Name: pubtune..titles (dbid=7, objid=208003772,lockscheme=Datapages)

 Page Locks     SH_PAGE                UP_PAGE               EX_PAGE
 ----------   ----------             ----------            ----------
 Grants:          94488                  4052                  4828
 Waits:             532                   500                   776
 Deadlocks:           4                     0                    24
 Wait-time:    20603764 ms          14265708 ms            2831556 ms
 Contention:       0.56%                10.98%                13.79%

*** Consider altering pubtune..titles to Datarows locking.
```

Output dow	Value
Grants	The number of times the lock was granted immediately.
Waits	The number of times the task needing a lock had to wait.
Deadlocks	The number of deadlocks that occurred.
Wait-times	The total number of milliseconds that all tasks spent waiting for a lock.
Contention	The percentage of times that a task had to wait or encountered a deadlock.

sp_object_stats recommends changing the locking scheme when total contention on a table is more than 15 percent, as follows:

- If the table uses allpages locking, it recommends changing to datapages locking.
- If the table uses datapages locking, it recommends changing to datarows locking.

Accessing Adaptive Server Data



- Adaptive Server has a multilayered approach for allowing an end user access to data on the server
  - The end user must have permission to log on to the server
  - Then, the end user must have permission to access a given database
  - Finally, the end user must have permission to use a given object

System Roles

- A system role is a set of privileges assigned to a given login
  - They allow the end user of that login to perform necessary administration and security tasks
- Adaptive Server system roles include:
  - System Administrator (SA)
  - System Security Officer (SSO)
  - Server Operator (OPER)
- A given login can have zero, one, or many roles
  - The more roles a login has, the more authority the login has

System Administrator (SA) Privileges

- Server privileges
  - Modify non-security configuration parameters
- Disk resource allocation privileges
  - Manage disk storage
  - Create user databases and grant ownership of them
- Access privileges
  - Grant and revoke the SA role to other logins
  - Grant permissions to server users
- Backup and restoration
  - Back up and restore any database
  - Back up and restore any transaction log
- System Management privileges
  - Shut down the server or kill a process (i.e. **kill**)
  - Use tools to diagnose system problems (e.g. **dbcc**)
- A login with **sa_role** is treated as dbo in every database, and grants and revokes are not enforced against it. **sa_role** can override security.
- **sa_role** is discussed in more detail later in this module

System Security Office (SSO) Privileges

- Access privileges
  - Create server logins and assign initial passwords
  - Modify logins
  - Change passwords
  - Set the password expiration interval
  - Create and manage user-defined roles
  - Grant the use of proxy authorization
  - Grant and revoke SSO and OPER roles to other logins
  - Manage the audit system
  - Lock and unlock logins
  - Drop logins

Server Operator (OPER) Privileges

- Backup and restore privileges
  - Back up and load all databases
  - Back up and load all transaction logs
- Database owners can back up and load databases and transaction logs for the databases they own

Viewing Roles for a Given Login

Syntax:

```
sp_displaylogin login_name
```

Example:

```
sp_displaylogin nyar
-
Suid: 4
Loginame: nyar
Fullname: Natasha Yar
Default Database: sybsecurity
Default Language:
Auto Login Script:
Configured Authorization: sso_role (default on)
Locked: NO
Date of Last Password Change: Oct 10 1999 8:44PM
Password expiration interval:
Password expired: NO
...
```

Generating Passwords for SSO Logins

- Difficulties can be encountered if all of the end users who have SSO role leave the company or forget their passwords
  - The only role that is authorized to change passwords for other users is the SSO role
- If this should occur:
  - Shut down the server
  - Add the -p option to the dataserver command in the RUNSERVER file
  - Restart the server
    - The **-p** command makes the server generate a new password for a login with the SSO role and displays the password to the console
  - Log in using the login with the SSO role
  - Reassign passwords and/or roles as needed

- Example:

```
#!/bin/sh
#
# Adaptive Server name:     SYBASE
# Master device path:       /work/ASE125/SYBASE_master.dat
# Error log path:           /work2/ASE125/ASE-12_5/install
#  SYBASE/log
# Directory for shared memory files: /work2/ASE125
#
/work2/ASE125/ASE-12_5/bin/dataserver \
-sSYBASE \
-d/work2/ASE125/SYBASEmaster.dat \
-e/work2/ASE125/ASE-12_5/install/SYBASE.log \
-M/work2/ASE125 \
-psa
```

- A user can change his or her password at any time

  Syntax:

```
sp_password caller_password, new_password [,
 login_name ]
```

  Example: (As William)

```
sp_password lefevre1, Zp_23222
```

  Example: (As sso)

```
sp_password iamsso, Zp_23222, William
```

Locking Logins

Syntax:

```
sp_locklogin [login_name, {"lock" | "unlock"} ]
```

Example:

```
sp_locklogin William, "lock"
```

- When a login is locked, the user cannot log on to the server
- Locking a login is more secure than dropping the login
  - When a login is locked, the server user ID (suid) value is retained, preventing it from being reused
  - When a login is dropped, the suid value can be reused
- With no parameters, returns a list of locked logins
- The last unlocked login cannot be locked with the SA role
  - Same for the last unlocked login with the SSO role

Viewing Information About Logins

Syntax:

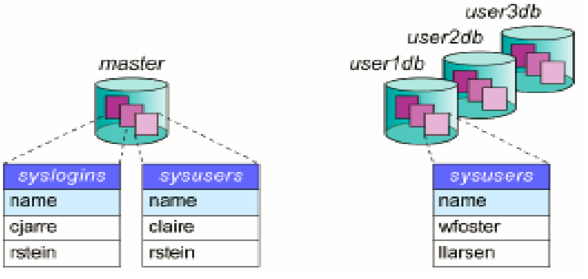
```
sp_displaylogin login_name
```

Example:

```
sp_displaylogin William
-
Suid: 57
Loginame: William
Fullname: William LeFevre
Default Database: salesdb
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Sept 7,2005 8:44PM
Password expiration interval: 0
Password expired: NO
...
```

Database Users

- To access a database, a given login must be listed as a user of that particular database
- Users are listed in the sysusers table in each database



Adding Database Users

Syntax:

```
sp_adduser login_name [, name_in_database [,
group_name ] ]
```

Examples:

```
sp_adduser William, William10
sp_adduser Cameron
```

- If no value is specified for *name_in_database*, the value for *login_name* is used for *name_in_database*

Viewing Information About Users

Syntax:

```
sp_helpuser [name_in_db ]
```

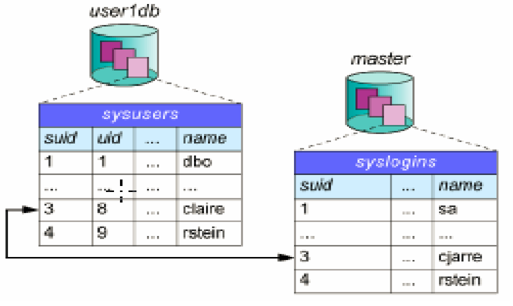
Examples:

```
sp_helpuser
-
Users_name ID_in_db Group_name Login_name
---------- -------- ---------- ----------
Sofia            16          public      Sofia
William10        17          public      William
Cameron          18          public      Cameron
```

- If you execute the procedure with a parameter, it returns information about the specified user

System Tables Related to Database Users

- Adding a user adds a row to sysusers



People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Learn more about us at

www.capgemini.com