SYBASE DBA Notes

Sybase ASE 15.0

Prepared by – Abhisek Vyas

Document Version 1.0

Team, SybaseRays.com

SybaseRays

What are database devices?

==========

A database device stores the objects that make up databases. The term device does not necessarily refer to a distinct physical device: it can refer to any piece of a disk (such as a disk partition) or a file in the file system that is used to store databases and their objects.

Each database device or file must be prepared and made known to Adaptive Server before it can be used for database storage. This process is called initialization.

After a database device has been initialized, it can be:

- Allocated to the default pool of devices for the create and alter database commands.
- Assigned to the pool of space available to a user database.
- Assigned to a user database and used to store one or more database objects.
- Assigned to store a database's transaction logs

==========

Diskinit:

==========

disk init divides the database devices into allocation units, groups of 256 logical pages. The size of the allocation unit depends on which logical page size your server is configured for **(2, 4, 8, or 16K)**.

Note: The logical device name is known only to Adaptive Server, not to the operating system on which the server runs.

Note: The number of database devices you can create is limited by the number of devices configuration parameter. Adaptive Server is initially configured for 10 devices.

Note: If you do not include a unit specifier for the size argument of disk init or disk reinit, size is measured, by default, in number of virtual pages. Thus, if you enter size = 15000, Adaptive Server assumes 15,000 virtual pages. A virtual page is 2048 bytes.

dsync flag: For devices initialized on UNIX operating system files, the dsync setting controls whether or not writes to those files are buffered. When the dsync setting is on, Adaptive Server opens a database device file using the UNIX dsync flag. The dsync flag ensures that writes to the device file occur directly on the physical storage media, and Adaptive Server can recover data on the device in the event of a system failure.

When dsync is off (false), writes to the device file may be buffered by the UNIX file system, and the recovery of data on the device cannot be ensured.

Turn off dsync only when data integrity is not required, or when the System Administrator requires performance and behavior similar to earlier Adaptive Server versions.

trad offs of dsync flag ON: If database device files on these platforms use the dsync option, the Adaptive Server engine writing to the device file blocks until the write operation completes. This can cause poor performance during update operations.

Alternative of dsync: In cases where highest write performance is required (but data integrity after a system failure is not required) turning dsync off yields device file performance similar to earlier Adaptive Server versions. For example, you may consider storing tempdb on a dedicated device file with dsync disabled, if performance is not acceptable while using dsync.

==========

SybaseRays

Alter database:

==========

Note: Each time you issue the alter database command, dump the master database.

==========

sysusage table:

==========

The database storage information is listed in **master..sysusages**. Each row in master..sysusages represents a space allocation assigned to a database. Thus, each database has one row in sysusages for each time create database or alter database assigns a fragment of disk space to it.

==========

Global variables:

==========

These global variables show page size information:

- @@maxpagesize – logical page size
- @@pagesize – virtual page size

==========

Adding logins:

==========

Adding new users

1 A System Security Officer uses sp_addlogin to create a server login account for a new user.

2 A System Administrator or Database Owner uses sp_adduser to add a user to a database. This command can also give the user an alias or assign the user to a group3 A System Security officer grants specific roles to the user.

4 A System Administrator, Database Owner, or object owner grants the user or group specific permissions on specific commands and database objects.

Users or groups can also be granted permission to grant specific permissions on objects to other users or groups.

Note: When you execute sp_addlogin, Adaptive Server adds a row to master.dbo.syslogins, assigns a unique user ID (suid) for the new user, and fills in other information. When a user logs in, Adaptive Server looks in syslogins for the name and password provided by the user. The password column is encrypted with a one-way algorithm so it is not human-readable.

==========

SybaseRays

Adding users in database:

==========

Note: sp_adduser adds a row to the sysusers system table in the current database.

* Discretionary access controls (DACs) allow you to restrict access to objects and commands based on a user's identity, group membership and active roles. The controls are "discretionary" because a user with a certain access permission, such as an object owner, can choose whether to pass

that access permission on to other users.

Adaptive Server's discretionary access control system recognizes the following types of users:

• Users possessing one or more system defined roles: System Administrator, System Security Officer, Operator, and other roles

• Database Owners

• Database object owners

• Other users

Permissions for creating databases:

==========

Only a System Administrator can grant permission to use the create database command. The user that receives create database permission must also be a

valid user of the master database because all databases are created while using master.

==========

Concrete identification:

==========

Adaptive Server identifies users during a session by login name. This identification applies to all databases in the server. When the user creates an object, the server associates both the owner's database user ID (uid) and the creator's login name with the object in the sysobjects table.

This information concretely identifies the object as belonging to that user, which allows the server to recognize when permissions on the object can be granted implicitly.

If an Adaptive Server user creates a table and then creates a procedure that accesses the table, any user who is granted permission to execute the procedure does not need permission to access the object directly.

For example, by giving user "mary" permission on proc1, she can see the id and descr columns from table1, though she does not have explicit select permission on the table:

```
create table table1 (id int,amount money,descr varchar(100))

create procedure proc1 as select id, descr from table1

grant execute on proc1 to mary
```

SybaseRays

There are, however, some cases where implicit permissions are only useful if the objects can be concretely identified. One case is where aliases and cross-database object access are both involved. You cannot drop an alias if the aliased login created any objects or thresholds.

Before using sp_dropalias to remove an alias that has performed these actions, remove the objects or procedures. If you still need them after dropping the alias, re-create them with a different owner.

==========

sp_configure "allow updates to system tables"

==========

To check current value:

```
1> sp_configure "allow updates to system tables"
2> go
 Parameter Name                Default    Memory Used  Config Value        Run Value  Unit    Type
 ----------------              -------    ------------ ------------        ---------  ----    -----
 allow updates to system tables    0          0            0                 0        switch  dynamic
```

To enable "allow updates to system tables":

```
1> sp_configure "allow updates to system tables", 1
2> go
 Parameter Name                Default    Memory Used  Config Value        Run Value  Unit    Type
 ----------------              -------    ------------ ------------        ---------  ----    -----
 allow updates to system tables    0          0            1                 1        switch  dynamic

(1 row affected)
Configuration option changed. ASE need not be rebooted since the option is dynamic.
Changing the value of 'allow updates to system tables' does not increase the amount of memory Adaptive Server uses.
(return status = 0)
```

allow updates to system tables enables users with the System Administrator role to make changes to the system tables and to create stored procedures that can modify system tables. A database administrator can update system tables in any tables that he or she owns if allow updates to system tables is enabled.

System tables include:

• All Sybase-supplied tables in the master database

• All tables in user databases that begin with “sys” and that have an ID value in the sysobjects table of less than or equal to 100

Stored procedures and triggers you create while allow updates to system tables is set on can update the system tables, even after the parameter has been set off.

SybaseRays

Because the system tables are so critical, Sybase suggests that you set this parameter to on only in highly controlled situations. To guarantee that no other users can access Adaptive Server while the system tables can be directly updated, restart Adaptive Server in single-user mode.

Note: Its recommended to follow following steps, before updating any system table.

- Modify system tables only inside a transaction. Issue a begin transaction command before you issue the data modification command.
- Verify that only the rows you wanted changed were affected by the command and that the data was changed correctly.
- If the command was incorrect, issue a rollback transaction command. If the command was correct, issue a commit transaction command.

Note: Triggers, defaults, and rules are not allowed in system tables. If you try to create a trigger or bind a rule or default to a system table, you will get an error message.

Example:

```
1> create trigger trh_1 on sysusers
2> for delete
3> as
4> select db_name()
5> go
Msg 2112, Level 16, State 1:
Server 'SYBASERAYS_SERVER_1', Procedure 'trh_1', Line 4:
CREATE TRIGGER failed on table 'sysusers' because it is a system table. Triggers
are not allowed on system tables
```

==========

System Procedures:

==========

The names of all system procedures begin with “sp_”. They are located in the sybsystemprocs database, but you can run many of them in any database by issuing the stored procedure from the database or by qualifying the procedure name with the database name.

Sybase-supplied system procedures (such as sp_who) are created using the installmaster installation script.

If you execute a system procedure in a database other than sybsystemprocs, it operates on the system tables in the database from which it was executed.

For example, if the Database Owner of pubs2 runs sp_adduser from pubs2 or issues the command pubs2..sp_adduser, the new user is added to pubs2..sysusers.

However, this does not apply to system procedures that update only tables in the master database.

The system procedures use several system procedure tables in the master and sybsystemdb databases to convert internal system values (for example, status

SybaseRays

bits) into human-readable format. One of these tables, spt_values, is used by a variety of system procedures

Creating System procedures:

System Administrators can write system procedures that can be executed in any database. Simply create a stored procedure in sybsystemprocs and give it a name that begins with “sp_”. The uid of the stored procedure must be 1, the uid of the Database Owner.

Most of the system procedures that you create query the system tables. You can also create stored procedures that modify the system tables, although this is not recommended.

To create a stored procedure that modifies system tables, a System Security Officer must first turn on the allow updates to system tables configuration parameter. Any stored procedure created while this parameter is set to “on” will always be able to update system tables, even when allow updates to system tables is set to “off.” To create a stored procedure that updates the system tables:

1 Use sp_configure to set allow updates to system tables to “on.”

2 Create the stored procedure with the create procedure command.

3 Use sp_configure to set allow updates to system tables to “off.”

System extended stored procedures:

An extended stored procedure (ESP) provides a way to call external language functions from within Adaptive Server. Adaptive Server provides a set of ESPs; users can also create their own. The names of all system extended stored procedures begin with “xp_”, and are located in the sybsystemprocs database.

One very useful system ESP is xp_cmdshell, which executes an operating system command on the system that is running Adaptive Server.

You can invoke a system ESP just like a system procedure. The difference is that a system ESP executes procedural language code rather than Transact-SQL statements. All ESPs are implemented by an Open ServerTM application called XP ServerTM, which runs on the same machine as Adaptive Server. XP

Server starts automatically on the first ESP invocation.

==========

Usually All Sybase related Script located at:

==========

/opt/sybase/ASE-15_0/scripts

Following scripts:

-rw-r--r--. 1 root root 45346 Jul 7 03:47 installpubs2

-rw-r--r--. 1 root root 471429 Jul 7 03:47 installpix2

-rw-r--r--. 1 root root 302315 Jul 7 03:47 ins_syn_sql

-rwxr-xr-x. 1 root root 3516 Jul 7 03:47 checksvrsuids

-rw-r--r--. 1 root root 54262 Jul 7 03:47 installpubs3

SybaseRays

```
-rw-r--r--. 1 root root  497699 Jul  7 03:49 instmsgs.ebf
-rw-r--r--. 1 root root   75516 Jul  7 03:49 uninstmsgs.ebf
-rw-r--r--. 1 root root   88551 Jul  7 03:57 installmontables
-rwxr-xr-x. 1 root root   32034 Jul  7 04:12 installupgrade
-rw-r--r--. 1 root root  100060 Jul  7 04:12 installsecurity
-rw-r--r--. 1 root root    2930 Jul  7 04:12 installmodel
-rw-r--r--. 1 root root 6098961 Jul  7 04:12 installmaster
-rw-r--r--. 1 root root  286939 Jul  7 04:12 installdbccdb
-rw-r--r--. 1 root root  257291 Jul  7 04:12 installdbccalt
-rw-r--r--. 1 root root   18391 Jul  7 04:12 installcommit
-rw-r--r--. 1 root root  282327 Jul  7 04:12 installpcidb
-rw-r--r--. 1 root root   68186 Jul  7 04:12 installmsgsvss
-rw-r--r--. 1 root root  601774 Jul  7 04:12 installjsdb
-rw-r--r--. 1 root root 1141969 Jul  7 04:12 installhasvss
-rw-r--r--. 1 root root  390705 Jul  7 04:12 installdbextend
```

==========

deadlock checking period

==========

The report displayed by sp_configure is constructed mainly from the master..sysconfigures and master..syscurconfigs system tables, with additional information provided from sysattributes, sysdevices, and other system tables.

You can query these system tables to get a complete list of current memory usage, as well as minimum, maximum, and default values, with the following query:

```
select b.name, memory_used, minimum_value,
maximum_value, defvalue
from master.dbo.sysconfigures b,
master.dbo.syscurconfigs c
where b.config *= c.config and parent != 19
and b.config > 100
```

deadlock checking period specifies the minimum amount of time (in milliseconds) before Adaptive Server initiates a deadlock check for a process that is waiting on a lock to be released. Deadlock checking is time-consuming overhead for applications that experience no deadlocks or very few, and the overhead grows as the percentage of lock requests that must wait for a lock also increases.

If you set deadlock checking period to a nonzero value (n), Adaptive Server initiates a deadlock check after a process waits at least n milliseconds.

For example, you can make a process wait at least 700 milliseconds for a lock before each deadlock check by entering:

SybaseRays

sp_configure "deadlock checking period", 700

If you set deadlock checking period to 0, Adaptive Server initiates deadlock checking when each process begins to wait for a lock. Any value less than the number of milliseconds in a clock tick is treated as 0. See “sql server clock tick length” for more information.

Configuring deadlock checking period to a higher value produces longer delays before deadlocks are detected. However, since Adaptive Server grants most lock requests before this time elapses, the deadlock checking overhead is avoided for those lock requests. If your applications deadlock infrequently, set deadlock checking period to a higher value to avoid the overhead of deadlock checking for most processes. Otherwise, the default value of 500 should suffice.

Use sp_sysmon to determine the frequency of deadlocks in your system and the best setting for deadlock checking period.

```
1> sp_configure "deadlock checking period"
2> go
 Parameter Name            Default    Memory Used  Config Value  Run Value    Unit         Type
 -------------------------  -------    -------     ------------  -------     ---------    --------
 deadlock checking period   500        0           500           500         milliseconds  dynamic

===========================================
```

SybaseRays