Little-Known New Features in ASE 12.5.0.1

By Rob Verschoor

Learning the secrets of the latest Sybase ASE release

In March 2002, ASE 12.5.0.1 was released. This release is the first major patch for 12.5 (actually, it's not just a "patch," but an "interim release," meaning it has undergone a very extensive test cycle). ASE 12.5.0.1 not only contains a large number of bug fixes, there are also some interesting new features. Unfortunately, some of these new features are not documented very clearly, or even at all, so they may well have escaped your attention. In this article I'd like to discuss some of these features so that they won't go unnoticed.

Please note that this article is not a complete overview of all new features in 12.5.0.1: See the document file named "$SYBASE/docs/newfunc.pdf" in the 12.5.0.1 distribution kit for more information about other new features.

Extensions to sp_monitorconfig

For DBAs, this is probably the most useful improvement in ASE 12.5.0.1: Additional configuration parameters supported by *sp_monitorconfig*. sp_monitorconfig displays how much of the configured value of a configuration parameter has been used since the last time ASE was started. For example:

```
1> sp_monitorconfig "open indexes"
2> go
Name                    num_free    num_active  pct_act     Max_Used    Reused
----------------------- ----------- ----------- ----------- ----------- -----------
number of open indexes  492         508         50.80       1000        Yes
```

This information is essential for tuning ASE configuration parameters: The above example shows that of the 1,000 available index descriptors ("number of open indexes"), at some point since the last reboot all have been in use. This means that the DBA should consider increasing this configuration parameter.

Before ASE 12.5.0.1, sp_monitorconfig supported only six configuration parameters. The big improvement in 12.5.0.1 is that another 28 configuration parameters have been added. Some of these additional parameters are relevant for every ASE DBA. For example, sp_monitorconfig will now report the highest number of locks, and the maximum amount of procedure cache used since the last reboot; these two pieces of information have been on my personal wish list for years, and I'm glad that this has now been implemented.

For the procedure cache, this information is essential for knowing whether the cache is too large or too small—there is just no other way to get this information (*sp_sysmon* reports some information about the procedure cache, but that provides only very indirect information about the procedure cache sizing).

You'll probably also see the benefit of knowing the highest of number of locks that's been used: suppose 100,000 locks have been configured—how do you know whether this is too many, or only just enough? Only sp_monitorconfig will give you the correct information.

Rob Verschoor is a freelance consultant in The Netherlands, specializing in Sybase ASE. He can be reached at rob@sypron.nl.

As mentioned, there are 28 new configuration parameters (34 in total) in sp_monitorconfig. You can display the complete list as follows:

```
1> sp_monitorconfig "all"
2> go
```

There really is no other way to get the information reported by sp_monitorconfig. I'm using it a lot, and I'm sometimes wondering how we all managed to do without this information for so long?

Lastly, you might wonder if sp_monitorconfig has an impact on ASE performance. I understand that performance tests by Sybase engineering have shown that there is no performance decrease (note that most of the monitoring performed by sp_monitorconfig can be disabled by booting ASE with traceflag 3631).

Client IP address

Did you know that the IP address of each client session is available in ASE since version 12.5? In case you didn't: a client's IP address is stored in *sysprocesses.ipaddr*. This IP address can be useful to find out which physical user corresponds to a certain client session (if this is new for you, you should also check out the new column *sysprocesses.loggedindatetime*—this contains the time when the client connected to ASE).

In ASE 12.5.0.1, the stored procedure *sp_client_addr* has been added to provide quick access to the IP address of a specific session:

```
1> sp_client_addr 17
2> go
spid        hostname    ipaddr
------      ----------  ---------------
17          pluto       1.0.0.107
```

When invoked without parameters, sp_client_addr will display all client sessions.

Small Business Edition and Developer's Edition

The new features in 12.5.0.1 are not only technical: There are also some interesting new developments on the commercial side. Starting with ASE 12.5.0.1, ASE comes in two flavors, the Enterprise Edition (EE), which is ASE as we've always known it, and the Small Business Edition (SBE).

The Small Business Edition is aimed at customers running production servers on less powerful hardware, and will have a lower price tag than the Enterprise Edition, making ASE more attractive for smaller systems. Corresponding to the lower license price, SBE may only be deployed on hardware with four CPUs or less. Also, SBE has some limitations built in: There can be no more than 256 user connections, no more than four ASE engines, and query parallelism is disabled. Also, some of the special options (at least High Availability and DTM) will not be available with SBE.

Another new ASE flavor will be released in the summer of 2002. Sybase will be offering a Developer's Edition of ASE, which is a very low-cost version of ASE 12.5.x. This can only be used for development and test activities, not for production. The Developer's Edition will be limited to one ASE engine and ten user connections, but will otherwise be a full-featured ASE version. Also, all special options like Java-in-ASE, High Availability, Distributed Transaction Management, file access via CIS (ASE_XFS), etc., will be included, so that developers can use these features to create applications. I think this Developer's Edition is good news, and it should certainly help to make ASE more popular.

Both the SBE and the Developer's Edition of ASE are expected to be available by the time this article goes to press. However, further information about price and license details were not yet available.

sp_sysmon Improvements

As every ASE DBA knows, *sp_sysmon* provides a lot of interesting data, but drawing conclusions from this data can be challenging. In 12.5.0.1, some changes have been made to sp_sysmon that may make it easier to interpret the output. Specifically, sp_sysmon will print "tuning recommendations" in its output, which suggest certain configuration changes. For example, when the spinlock contention for a cache is more than 10%, sp_sysmon will suggest splitting the cache into multiple caches (only when the server is running with two or more engines). sp_sysmon may also suggest changing the number of engines, allowing more housekeeper activity, or adjusting other configuration parameters.

The good thing about this change is that DBAs don't have to reread *The Performance & Tuning Guide* to interpret some of the sp_sysmon data, because the interpretation has been hard-coded. However, blindly following the system's recommendations does not seem a good idea, because they are based on just a single sp_sysmon session; for example, during off-hours, sp_sysmon might suggest to offline some engines, while it wouldn't suggest this during peak hours. It is still up to the DBA to judge whether these recommendations make sense.

Traceflag 243: Do Not Expand select *

Back in ASE 12.0, a change was implemented in the way **select *** is handled when creating compiled objects (stored procedures, view, triggers). In the T-SQL source code, **select *** was expanded into the full list of columns. Prior to 12.0, **select *** remained unchanged. Note that this change doesn't make any difference for the compiled object itself—the difference is in the T-SQL source text which is stored in the *syscomments* table. This text does not normally fulfill any function, but may be used internally when performing an upgrade of ASE (one reason for the modification was to avoid certain types of errors when performing an upgrade).

Now, in ASE 12.5.0.1, traceflag 243 has been introduced to revert to the old behavior (i.e., leave **select *** unchanged). Why should you bother about all this? Well, you should pay attention to this issue in the following cases:

- You're currently running ASE 11.9 or earlier and you'll be upgrading to 12.0 or later.
- You're using the syscomments table in ASE as a source code repository for your compiled objects, i.e., *defncopy* (or other tools accessing syscomments) is used to regenerate the T-SQL source text. While it is probably not a good idea to use ASE for this, it often happens in practice.

In either of these cases, the above story is actually pretty important for you, because it may affect the functionality of objects that are recreated from this source code. Consider the following example:

```
1> create table t (a int, b int)
2> go
1> create procedure p1 as select * from t
2> go
1> dbcc traceon(243)
2> go
1> create procedure p2 as select * from t
2> go
```

As expected, the two procedures have identical functionality. Now, check the text of both procedures with *sp_helptext*. Then, add another column to the table as follows:

```
1> alter table t add c int null
2> go
```

When you now drop both stored procedures and then recreate them using the T-SQL text shown by sp_helptext, you'll find that the recreated stored procedures behave differently: "p1" will show only the original two columns "a" and "b," while the recreated procedure "p2" will show all three columns including the new column "c".

It is exactly this type of problem that may cause trouble when performing ASE upgrades, and this was the reason for implementing the described change in 12.0. However, depending on your situation, you may prefer the pre-12.0 behavior, which you get by using traceflag 243.

(Note that the issue described here does not directly affect a migration from 11.9 to 12.0/12.5, but it may start to play a role after that migration, as the default functionality has changed in comparison to pre-12.0.)

New Config Parameter *Number of Histogram Steps*

When running the **update statistics** command, a DBA can optionally specify the number of histogram steps to be used in the statistics for that table. In principle, when using more steps, the statistics will be more accurate, and bad query plans will occur less often. By default, the number of histogram steps is 20, and this has resulted in headaches for many DBAs who wanted to use a higher number of steps. One common problem was that the setting for the number of steps for a specific table reverted to the default of 20 once the table was dropped and recreated. In 12.5.0.1 (as well as in 12.0.0.4), the new configuration parameter *number of histogram steps* has been added to address such problems. This new parameter lets a DBA specify the default number of steps on a server-wide basis.

SQL Advantage is Back!

Were you one of those ASE users who upgraded to ASE 12.0 and couldn't find SQL Advantage in the PC client environment anymore? If so, this wasn't your fault—SQL Advantage was removed from 12.0 for unknown reasons (but if I may make a guess, I'd think that jisql, a Java-GUI query tool, was expected to replace SQL Advantage). Following many user complaints, SQL Advantage has been re-introduced in 12.5.0.1. It is now located in the directory %SYBASE%\sqladv-12_5. If you're not on 12.5.0.1 yet, but you would still like to have SQL Advantage back, simply copy the 11.9 executable from %SYBASE%\bin\sqladv.exe to your 12.0 or 12.5 environment—it will work fine.

Java-in-ASE: Improved Performance

The Java Virtual Machine (JVM) in ASE is reported to have undergone significant performance improvements. Because I couldn't find any "official" figures quantifying this, I've done some performance tests myself, using some simple Java-in-ASE functionality. While these tests were certainly not full-blown benchmarks, there seems to be a significant

performance improvement indeed: In my tests, I measured an improvement of about 100% (twice as fast). Please note that these performance improvements are purely internal to the JVM and ASE; there is no change in functionality.

ASE Replicator

Many of us are familiar with cases where some simple data replication functionality would very useful, but the price tag of Sybase Replication Server prohibits implementation. There's good news, because ASE 12.5.0.1 includes a new ASE feature called ASE Replicator which can be used for implementing lightweight data replication. The big advantage of ASE Replicator is that it's free: It is an ASE feature which comes with ASE at no additional cost.

ASE Replicator supports ASE-to-ASE, log-based, publish/subscribe replication of DML and stored procedures. However, ASE Replicator is not suitable for high workloads, nor does it have all the features of Replication Server. While the underlying principles of ASE Replicator are very similar to those of Sybase Replication Server, they're different products for different types of replication requirements: ASE Replicator is really a "lightweight" replication solution, for low-volume, non-critical replication; in contrast, Replication Server is the product needed for enterprise-level replication. ASE Replicator should therefore not be seen as a replacement for Replication Server.

More information about ASE Replicator is available at www.sypron.nl/aserep.html.

ddlgen: Reverse-Engineering Tool

ddlgen is a command-line tool to reverse-engineer the schema of database objects (i.e., generate the T-SQL statements needed to recreate those objects). Though ddlgen was new in 12.5, and only a few additional options were added in 12.5.0.1, I want to mention it anyway: ddlgen still seems to be quite unknown, even though it is a very useful tool for any DBA.

Not only can ddlgen reverse-engineer database objects, it also handles many server-level ASE objects, such as logins, devices and caches. One of the strong points of ddlgen is that it also works for pre-12.5 ASE versions like 12.0 and 11.9. I recommend all DBAs look at ddlgen!

ddlgen is located in %SYBASE%\Sybase Central 3.2 (Windows) or $SYBASE/sybcent32 (Unix). ddlgen is documented in the Utility Guide in the ASE 12.5 documentation.

sybmigrate: A New Migration Tool

ASE 12.5.0.1 includes a migration tool called *sybmigrate* (located in $SYBASE/ASE-12_5/bin). With only a few mouse clicks, sybmigrate copies an entire database, including schema and data, from one ASE server to another. The functionality brought by sybmigrate would probably be welcomed by many DBAs; unfortunately, it only works on ASE 12.5.0.1 or later, severely limiting its usefulness. I think it would therefore be great if sybmigrate were enhanced to support cross-version migration as well (e.g., from 11.9 or 12.0 to 12.5 or later). I have added ISUG enhancement request e01_207 for this, so if you think this would be useful to you, please vote for this request at www.isug.com.

Technically, sybmigrate migrates a database by using the ddlgen tool (see previous section) to reverse-engineer the schema from the source database, recreate the schema in the target database, and finally transfer the data using CIS (note that this is how you could perform a cross-version migration manually as well, because ddlgen also appears to work on pre-12.5 ASE versions).

The background of sybmigrate is the requirement to migrate databases between 12.5 servers with different server page sizes, because dumps cannot be loaded into a server with a different page size.

One of the strong points of sybmigrate is its user-friendliness: It takes only a small amount of user action to migrate an entire database, no matter how many tables it contains. By default, sybmigrate pops up a GUI, but it can also be used via a command-line interface.

When working with sybmigrate, I found an unexpected bonus: Although the sybmigrate documentation states that cross-platform migration is not possible, I have migrated databases between ASE 12.5.0.1 servers on NT and Linux without any problem.

sybmigrate has many different options which cannot be discussed in detail here. Detailed sybmigrate documentation is in Chapter 10 of the document $SYBASE/docs/newfunc.pdf. See $SYBASE /docs/sybmigrate/MT-quickstart.txt for some suggestions to get started.

disk resize Command

Have you ever wanted to make an existing database device larger (your master device, for example)? Until now, this was not directly possible; it could only be done with tricky workarounds. ASE 12.5.0.1 now supports the new command **disk resize**, which lets you enlarge an existing device. This example makes device "dev1" 100Mb larger:

```
1> disk resize name = "dev1", size = "100M"
2> go
```

disk resize works for both filesystem devices and raw devices (for raw devices, the OS must support enlargement)

Grant/Revoke for dbcc Commands

ASE 12.5.0.1 introduces the commands **grant dbcc** and **revoke dbcc**. As the name suggests, these commands grant and revoke execute rights on dbcc commands for a database to users or roles. This can be useful in situations where the task of performing dbcc checks is delegated to a specific user; using this new functionality, that user does not need to have *sa_role* anymore.

Note that only a subset of the officially documented dbcc commands can be granted; the supported dbcc commands are **checkalloc**, **checkcatalog**, **checkdb**, **checkstorage**, **checktable**, **checkverify**, **fix_text**, **indexalloc**, **reindex**, **tablealloc**, **textalloc**, **tune**. For example:

```
1> grant dbcc checkstorage on prod_db,
2> dbcc checkverify on prod_db to dbcc_role
3> go
```

Needless to say, DBAs should be careful granting permissions on these dbcc commands, as their execution may affect overall system performance.

More Databases per Transaction

ASE has always had a hard limit for the maximum number of databases that can be involved in one transaction: Before 12.5.0.1, this maximum was 16. In 12.5.0.1, the limitation has been removed. By configuring sufficiently high values for the configuration parameters *number of open databases*, *txn to pss ratio*, and *number of aux scan descriptors*, an unlimited number of database may be opened in one transaction.

Actually, a limitation still exists—the upper limit is the maximum number of databases in one ASE server—but as this value is 32767, I think that should be enough for most purposes.

Conclusion

As you can see, there is a lot of new stuff in 12.5.0.1. I hope these new features will be helpful for DBAs and improve the usefulness of ASE. ■