

Upgrading to ASE 15

An in-depth view of the (un)expected

Rob Verschoor
Senior Technology Evangelist
Sybase Engineering
robv@sybase.com

What we'll be talking about



Upgrading to ASE 15

- How to do it
- Things you must pay attention to
- Things to be aware of
- Things that may require some 'extra attention'
- Examples from actual customer migrations

What we won't be talking about...



What we'll be talking about



Structure of the morning session:

- Rob V (Sybase): Various aspects of ASE 15 migration
- Coffee break
- Stephen Longford (JP Morgan Chase) : JPMC's ASE 15 upgrade
- Steve Allen (Sybase UK): ASE 15 upgrades as seen by TechSupport
- Q & A

Why migrate to ASE 15?

Why migrate to ASE 15?



ASE 15 has many appealing new features

- New query processing infrastructure
 - ✓ Can deliver significant performance improvements
- Encryption
- Semantic partitioning
- Enhancements around update statistics

- Major XML enhancements
- Computed Columns & Function Indexes
- SQL User-Defined Functions
- Instead-of Triggers
- Application tracing
- Row-level locking on system tables
- ...and much, much more...

Why migrate to ASE 15?



ASE 15 seeks to address customer requirements around:

- The data explosion
 - ✓ Support larger database sizes (up to 1 Exabyte/ASE server)
 - ✓ Better support for mixed workload and DSS (new QP)
 - ✓ Better support for large-scale OLTP (semantic partitions)
- Availability
 - ✓ Reduce maintenance window (semantic partitions, datachange())
- Security
 - ✓ Column encryption
- Handling unstructured/semi-structured data
 - ✓ XML
 - ✓ Various developer features (CPC, SQL UDF, IOT, ...)

ASE 15 Query Processing



ASE 15

- Current latest version: 15.0.2 esd#2
- EOL for 12.5 is 31-Dec-2009
 - ✓ As always with future references: no guarantees!

Recommendation:

- Please get the latest ASE 15 patch if you run into any issue

Incidentally...

- The ASE shared disk cluster release (coming in 2008) is really a different thing -- let's not consider that here
 - ✓ A better high-availability solution suitable for specific classes of apps
 - ✓ Initial version will likely be "15.0.1 Cluster Edition"

ASE 15 upgrade success story: Western & Southern Financial Group



Customer

- Insurance, investments
- Fortune 500, \$47bn in assets
- Based in Cincinnati, Ohio



Project

Migration of ASE 12.5 servers on Solaris to ASE 15 on Linux

Results

- Performance gains averaging 83%
 - End-users initially suspicious of the system—it was too fast to be credible!
- Consolidated 8 servers to 3
- Achieved significant cost savings by using Linux and commodity hardware
- No problems during migration/upgrade

ASE 15 upgrade success story: Caixa Econômica Federal (Brazil)



Customer

- Banking; 100% owned by Brazilian Federal Government
- Main business: lending (as instrument of public policy)
- 2006: 4.6 billion banking transactions; 19,951 points of service; 33.6 million customers; R\$ 50.2 billion in deposits



Project

- Migrate Risk Management System (loan contract evaluations) to ASE 15
- Deploy semantic partitioning to reduce administrative window for 1TB database (growing 40Gb/month)

Results

- Time to update statistics for all tables went from 6 days to 28 hours
- Time for database backup went from 1 day to 1 hour
- Time for loading database dump reduced by 75%
- All queries 10-30% faster than in ASE 12.5.3

ASE 15 upgrade success story: eSpeed



Customer

- Electronic marketplaces and related trading technology for global capital markets
- Based in New York City



Project

Replace "Polling-Broadcasting" legacy solution with integrated real-time messaging (RTDS 4.0) in ASE 15

Results

- Eliminated database polling
- Improve timeliness of decision making
- Information flows that are faster, more relevant and actionable
- Eliminated need to write custom code to capture events
- Legacy systems brought into mix for more accurate view of business processes

How to upgrade to ASE 15

Upgrade Methodologies



Upgrading to ASE 15 is no different from other ASE versions

Upgrade In-Place

- Server is upgraded in place using the **upgrade** utility
- If upgrade fails, contact Sybase TS first
 - ✓ Upgrade utility automates steps – if it fails, TS may be able to get you past the point of failure and help manually complete the upgrade.

Traditional Dump/Load (new server)

Quiesce/Mount method

- Slight twist on dump/load but much faster
- Caveats/limitations
 - ✓ Database upgrades are at a device level, so unless databases are aligned on device boundaries, you may have to upgrade all the databases on the same devices.

Before you upgrade...



- Run the **pre-upgrade** tool
 - ✓ Fix any issues reported
 - ✓ Run **pre-upgrade** again... repeat...

- System databases increase in size slightly (4MB or so)
 - ✓ Add space before the upgrade
 - ✓ Minimum sizes for 2K server
 - master → 13MB, sybsystemprocs → 124MB

Upgrade Considerations



Update Statistics → run after upgrade

- Not required, but highly recommended (if old stats are around)...
 - ✓ ...in ASE 12.5, we had one algorithm for group by, etc. – no chance to make a costing error.
 - ✓ Now we have 2-3 choices for just about any group/sort/join strategy
 - it is more important that statistics are accurate to allow the best strategy to be picked
- Consider deleting all existing stats first (**delete statistics...**)
- ... then re-run **update [index] statistics**

Also, see separate slides about **update statistics** (later)

Upgrade Considerations



Boot Time Trace Flags

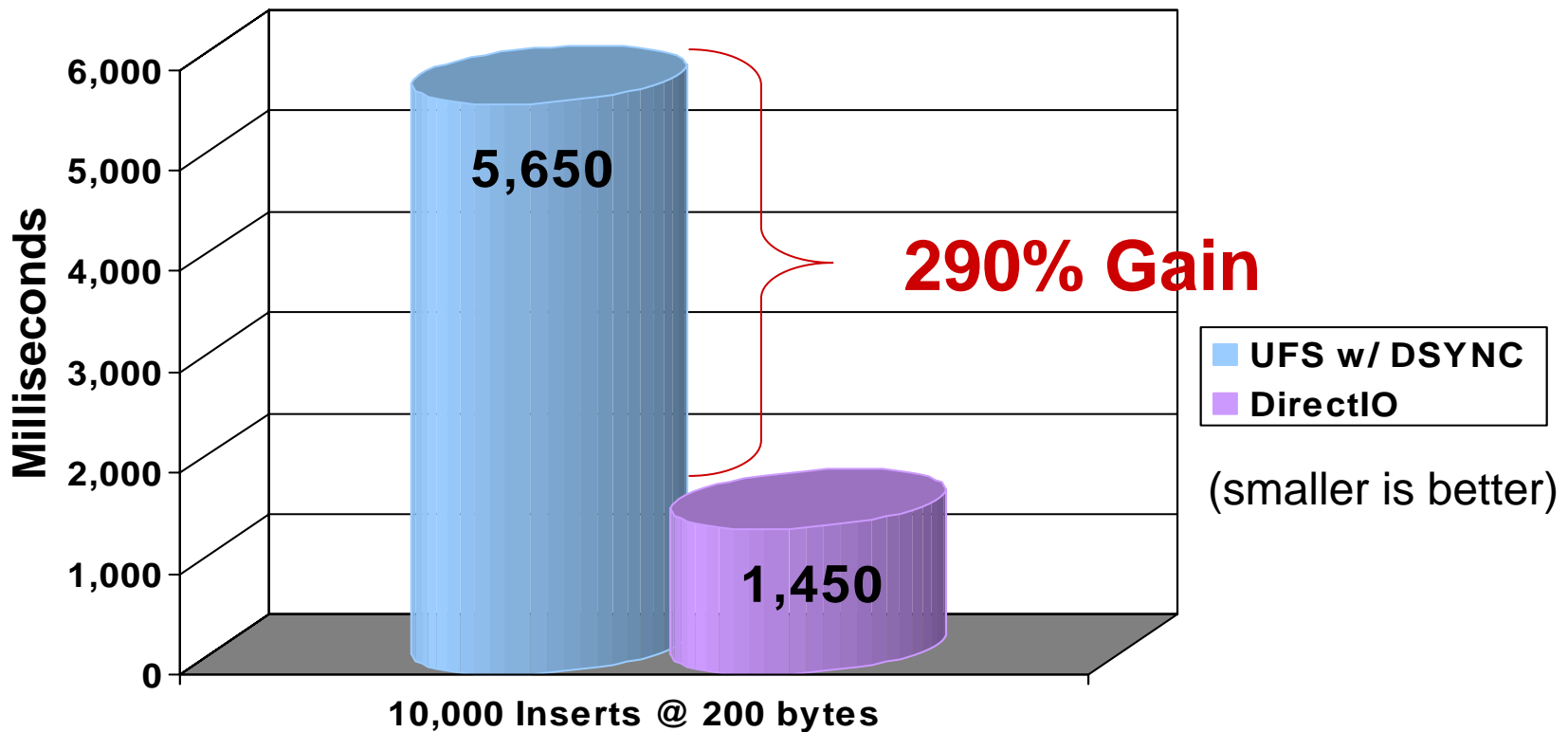
- Trace flags 291, 333, 364, 370, 396 no longer supported
- If booting with a trace flag to change optimizer behavior, it is likely that you will need to stop using it
 - ✓ Call Sybase Tech Support with specific trace flag questions

If you're using UFS DB devices...



Switch from DSYNC to DIRECTIO

- They are mutually exclusive
- **directio** may improve performance significantly



Tempdb & UFS



UFS is not always the best choice for tempdb...

- On some platforms, the OS implementation may not allow AIO to file system (HP-UX)
- ASE 12.5.0.3+ Implements "Lazy I/O"
 - ✓ Physical I/O's are deferred – skipped entirely if possible
 - ✓ Eliminates some of the need for tmpfs, etc.

...but applications may still benefit

- Make sure to test at full user load
- Make sure that **dsync** is off!
 - ✓ Common problem with tempdb on UFS
 - ✓ Manifested during ORDER BY, GROUP BY and other operations
- Test with **directio** enabled

Tempdb & UFS Tips



Use multiple tempdb's:

- One tempdb with **dsync** off and **directio** off
 - ✓ Good for read intensive operations
 - Batch reporting with large temp tables scanned for joins/index builds
- Another tempdb using raw devices or UFS-with-**directio**
 - ✓ Good for write-intensive operations

Make sure you have enough UFS cache

- If not using **directio**, you will use UFS cache no matter if **dsync** is on or off
- ASE + UFS cache size < 85% of physical memory

Make sure UFS cache is constrained

- Use OS kernel params to constrain it to prevent swapping

SySAM 2.0



ASE 15.0 comes with new version of license mgmt: SySAM 2.0

- You cannot avoid this in ASE 15
- You'll always need a license key in ASE 15
 - ✓ If not, ASE won't boot, or will shut down after a grace period

It ain't as bad as it sounds though...

- Sybase Tech Support is your friend here (aren't they always?)
- SySAM 2.0 also allows you to set up a redundant license servers for HA systems

Things you must pay attention to when migrating to ASE 15

ASE 15 Query Processing



#1 feature: new Query Processing infrastructure in ASE 15

- New query optimizer
- New query execution engine

You cannot avoid the new QP aspects of ASE 15

There is no trick or traceflag to switch back to the 12.5 optimizer!

✓ Honestly!

ASE 15 Join Methods



ASE 15 supports four join methods:

- Nested-Loop Join (NLJ)
 - ✓ Same as in pre-15.0

- N-ary NLJ
 - ✓ A variant of NLJ patented by Sybase

- Merge Join (MJ)
 - ✓ Strongly improved from ASE 12.0 implementation

- Hash Join (HJ)
 - ✓ Hash-based join method

Optimization Goals in ASE 15



Central concept in ASE 15 QP: "Optimization Goal"

- In essence, a hint to the query optimizer about the nature of the query being optimized, thus allowing the optimizer to be resource-efficient

Existing optimization goals:

- **allows_oltp** - uses a limited number of optimization criteria to find a good query plan aimed at OLTP
 - ✓ nested loop join, no parallelism
- **allows_mix** - (default) generates plans for mixed workloads
 - ✓ basically, **allows_oltp** + merge joins, parallelism
- **allows_dss** - generates optimal plans for high-complexity DSS
 - ✓ basically, **allows_mix** + hash joins

Query Processing Enhancements



Performance improvements for DSS / complex queries / aggregates / star schemas:

- Use **allows_dss** or **allows_mix**

Performance improvements for OLTP:

- Use **allows_oltp**
- Use partitions

In principle, there's no need to change any application SQL when upgrading to ASE 15

- Some specific exceptions will follow
 - ✓ Mostly apply to tools, not to business applications

Optimization Goals in 15.0



Optimization goal can be set at server, session and query level:

- Server:

```
sp_configure "optimization goal", 0, "allows_dss"
```

- Session (can be done in a login trigger!):

```
set plan optgoal allows_dss
```

- Query:

```
select * from A order by A.a  
plan "(use optgoal allows_dss)"
```

Optimization Goals in 15.0



Objective with optgoals should be: Keep It Simple

- Try using as little fine-tuning as possible
 - ✓ Ideal is just one server-wide setting and no lower-level overrides
 - ... this ideal is unlikely to give optimal performance in real life

- Some applications or some queries might benefit significantly from using a different optgoal than the server-wide setting...
 - ✓ Need to balance optimal performance tuning vs. required effort

ASE 15 Query Processing



Some practical migration advice:

- As a baseline, use **allrows_oltp** as your server-wide optgoal (closest to 12.5)
 - ✓ This won't get you most of those marvellous performance improvements, however (use **allrows_dss** for that)
 - ✓ Start with most defensive approach
- Don't use parallel query processing in ASE right off the start
 - ✓ ASE 15 has a fundamentally different approach towards parallelism than 12.5
 - ✓ Try serial mode first and see what it gets you
 - ✓ Determine where parallelism applies later
- Use MDA tables to find queries running slower than expected

ASE 15 Query Processing



Why not set the server-wide goal to 'allrows_dss' ?

- Because **allrows_dss** may use significantly more resources than **allrows_oltp**
 - ✓ 'Resources' = memory (procedure cache), compile time
- Because your workload may not benefit from all those access methods allows by **allrows_dss**

Optimization Goals in 15.0



To do some experimentation with different optgoals:

- Change server-level setting
 - ✓ Reboot, or run `sp_recompile` on all your tables
(use `sp_exec_table` → www.sypron.nl/exectab.html)
- If things get faster → good!
- If things get slower → try to find out which ones
 - ✓ Use MDA tables, run those queries with a different optgoal

Optimization Goals in 15.0



To do some experimentation with different optgoals:

- In general, your options are:
 - ✓ Use a server-level optgoal that's best for most queries -- override this for individual session or queries that benefit from a different optgoal
 - Better performance overall, but more work for you!
 - ✓ Use a server-level optgoal that doesn't cause any queries to run unacceptably slow, and do not do any fine tuning elsewhere
 - Simpler; less work, but also less benefit from ASE 15's capabilities

ASE 15 migration



Please, do all this testing before migrating your production system!

ASE 15 Query Processing



Optimization goals are really high-level names for a collection of 'optimization criteria' settings

- Optcriteria govern whether the optimizer may consider using a particular access method or algorithm
 - ✓ set hash_join { on | off }
 - ✓ set store_index { on | off }
 - ✓ set advanced_aggregation { on | off }
 - ✓ set merge_union_distinct { on | off }
 - ✓ ... and another 20 or so....

- Optcriteria are about advanced fine-tuning of optimization
 - ✓ Not recommended for your ASE 15 migration
 - ... unless you need it (as advised by Tech Support)
 - ... or you have time left

ASE 15 Query Processing



Advanced optimizer tuning options

- Resource granularity settings
- Optimization timeout limits

- Same advice: Don't go there unless advised by Tech Support

Things to keep in mind when migrating to ASE 15

Things to keep in mind for ASE 15 migration



New system table: syspartitions

- Pre-15 **syspartitions** is renamed **syslices** and otherwise unused
 - ✓ Pre-15 partitioned tables are unpartitioned during the 15.0 upgrade
- **syspartitions** contains essential storage-related data that was stored in **sysindexes** in pre-15:

<u>sysindexes</u>	<u>syspartitions</u>
first	firstpage
root	rootpage
doampg	datoampage
ioampg	indoampage

Things to keep in mind for ASE 15 migration



syspartitions now is a vital storage-related table

- Important: configure '**number of open partitions**' high enough
- Use **sp_monitorconfig 'all'** to find out if it's configured high enough
- Run **sp_countmetadata 'number of open partitions'** for an initial estimate
- **sysindexes** columns remain, but values will be 0 or unreliable

Things to keep in mind for ASE 15 migration



You don't need to change any SQL when upgrading to ASE 15, except when...

... calling the storage space built-ins:

- rowcnt()
- data_pgs()
- ptn_data_pgs()
- reserved_pgs()
- used_pgs()

Things to keep in mind for ASE 15 migration



Built-in functions related to sysindexes have changed

Pre-15

rowcnt(doampg)

data_pgs(objid, d/ioampg)

ptn_data_pgs(objid, ptnid)

reserved_pgs(objid, d/ioampg)

used_pgs(objid, doampg, ioampg)

15

row_count(dbid, objid [, ptnid])

data_pages(dbid, objid [, indid [, ptnid]])

data_pages(dbid, objid [, indid [, ptnid]])

reserved_pages(dbid, objid [, indid [, ptnid]])

used_pages(dbid, objid [, indid [, ptnid]])

Things to keep in mind for ASE 15 migration



Existing code keeps running, but prints a warning for every call of a pre-15 built-in:

The built-in function 'rowcnt' has been deprecated. Use the new built-in function 'row_count' instead. Deprecated function returns value of 0.

Things to keep in mind for ASE 15 migration



You don't need to change any SQL when upgrading to ASE 15, except when...

...doing direct queries against:

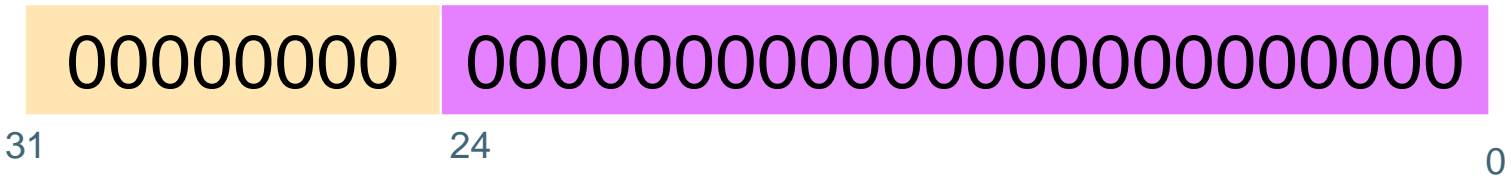
- sysdevices
- sysusages

Storage in ASE Pre-15.x



256 Device limit per server/32GB per device

- Limit of #device fragments per DB was removed a long time ago
- Virtual page was a 4 byte mask
 - ✓ Most significant byte = vdevno = 256 devices
 - ✓ Last 3 bytes = vpageno = $2^{24} * 2Kpg = 32GB$
 - Theoretically, a 16K server *could* have had larger devices, but due to the default 2K page, a 32GB limit was imposed.



High 8-bits represent the device number.

Low 24-bits represent the block number (2K offset within the device)

System Table Changes



New column sysdevices.vdevno

- **Low** and **high** columns no longer contain any info related to the device number

New column sysusages.vdevno

- **vstart** column no longer maps to device number

Relationship between sysdevices **and** sysusages **is now direct** **using the** vdevno **column**

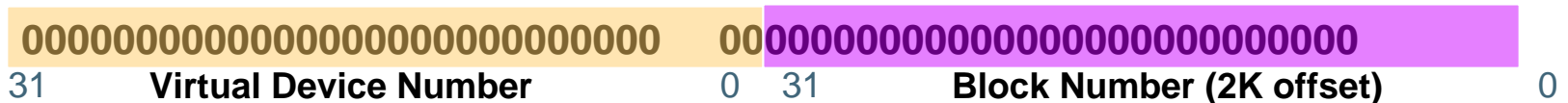
- No more: ... **where** sysusages.vstart **between** sysdevices.low **and** sysdevices.high

VLDB enhancements



<i>Attribute</i>	<i>Old (12.5.x) Limit</i>	<i>New (15.0) Limit</i>
Number of devices	256	2,147,483,648 (2³¹)
Maximum device size	32 Gb <i>32GB ← 2²⁴ * 2K vpg</i>	4 Tb
Maximum database size (2K / 4K / 8K / 16K)	4 Tb / ~8 Tb / ~ 8 Tb / ~8 Tb <i>8TB ← 256 devices * 32GB</i>	4 Tb / 8 Tb / 16 Tb / 32 Tb

In ASE 15.0, the virtual page number is represented by two 32-bit values. One is the device number, the other is the block number



Mapping Databases to Devices



Query to find device name for each database fragment

Old Way:

```
select u.lstart, d.name
from sysusages u, sysdevices d
where u.vstart >= d.low
and u.vstart <= d.high
and u.dbid = <database id>
```

New Way:

```
select u.lstart, d.name
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and u.dbid = <database id>
```

Vdevno Rules



- In pre-15, the vdevno passed to disk init must be less than the configured 'number of devices'
- In 15.0 it can be any value up to 2,147,483,647 independent of the current configured number of devices

This may or may not be useful for customers

- Ex: customers can group ASE virtual devices on the same physical disk by using the vdevno
 - ✓ vdevno 1000 to 1999 on physical disk 1
 - ✓ vdevno 2000 to 2999 on physical disk 2
- This sort of “coding” can be also be used for databases as well
 - ✓ Vdevno 1000 to 1999 belong to db A
 - ✓ Vdevno 2000 to 2999 belong to db B

VLSS in ASE 15.x



ASE 15.0 VLDB enhancements

- Devices = 2^{31} (2 Billion)
- Maximum device size = 4TB
- Databases/server = 32,767 (still)
 - ✓ Practical limit is probably still 100

The maximum storage will be:

- Database size:
 - ✓ 2^{31} pages * 16KB pg = 32 TB
- Theoretical server size:
 - ✓ 2^{31} devices * 4TB size → 8,589,934,592 TB
- Theoretical DB storage
 - ✓ 32,767 DB's * 32TB = 1 EB (exabyte) = 1,048,544 TB

Things to keep in mind for ASE 15 migration



Object ID has been replaced by partition ID in page header

Pre-15:

PAGE HEADER:

Page header for page 0x214D1000

pageno=617 nextpg=0 prevpg=0 **objid=1010786989** timestamp=0000 00000c30

nextrno=0 level=0 indid=0 freeoff=32 minlen=6

page status bits: 0x81 (0x0080 (PG_FIXED), 0x0001 (PG_DATA))

ASE 15:

PAGE HEADER:

Page header for page 0x22996800

pageno=857 nextpg=0 prevpg=0 **ptnid=1229687001** timestamp=0000 000038d3

nextrno=0 level=0 indid=0 freeoff=32 minlen=6

page status bits: 0x81 (0x0080 (PG_FIXED), 0x0001 (PG_DATA))

Things to keep in mind for ASE 15 migration



To determine the object for a given partition ID:

```
select object_name(id)
from syspartitions
where partitionid = ptnid
```

- As of 15.0.1, `partition_object_id(ptnid [, dbid])` can be used instead:

```
select object_name(partition_object_id(ptnid))
```

Things to keep in mind for ASE 15 migration



A partitioned table in pre-15 will be unpartitioned after upgrading to 15

- Each partition has its own partition ID in the page header
 - ✓ Retaining the partitions would imply rewriting all pages
- The pre-15 syntax for partitioning remains valid
 - ✓ Same as 'roundrobin' partitioning syntax
 - ✓ Only exception: **alter table...unpartition** is no longer allowed when the table has an index
- NB: actually, unpartitioned tables don't really exist anymore in 15
 - ✓ Each table has at least 1 roundrobin partition

sysdevices / sysusages



Impact of changes to sysdevices / sysusages

- sql or stored procedures that determined device space usage will no longer return correct results.
 - ✓ Most likely you will see a large negative number for space available
- **sp_helpdevice** has been altered to show free space on database devices
 - ✓ Can use code from **sp_helpdevice** as a model for new custom stored procedures.

sp_helpdevice



```
1> sp_helpdevice dev1
```

```
2> go
```

device_name	physical_name	description		status
	cntrltype	vdevno	vpn_low	vpn_high

dev1	[...] physical disk,	900.00 MB,	Free: 235.00 MB	16386
	0	2	0	460799

```
(1 row affected)
```

sp_helpdevice



```
1> sp_helpdevice dev1
```

```
2> go
```

device_name	physical_name	description		status	
	cntrltype	vdevno	vpn_low	vpn_high	

dev1	[...] physical disk,	900.00 MB,	Free: 235.00 MB	16386	
	0	2	0	460799	

```
(1 row affected)
```

dbname	size	allocated		vstart	lstart

z1	350.00 MB	Sep 20 2005	8:01:56:346PM	0	0
tempdb	4.00 MB	Sep 27 2005	2:23:56:430PM	307200	2048
demo3	250.00 MB	Jan 5 2006	6:53:26:643PM	179200	0
tempdb	6.00 MB	Apr 4 2006	7:34:55:430PM	309248	4096
demo	5.00 MB	Jun 7 2006	1:02:09:543PM	312320	0
demo2	50.00 MB	Jun 7 2006	1:02:10:543PM	314880	0

Things to keep in mind for ASE 15 migration



You don't need to change any SQL when upgrading to ASE 15, except when...

... using a non-ANSI approach towards order-by

Things to keep in mind for ASE 15 migration



Always use an order by clause when returning data to the client

- ...but you already did, right?
- The new ASE 15 execution engine is -internally- stream-oriented rather than step-by-step oriented
 - ✓ This means a different internal data flow, with possible implications for result set order

Group by ordering



The ANSI SQL standard specifies that result set ordering is undefined without an 'order by' clause

- Since 11.5 (parallelism) and 11.9 (DOL row forwarding), `order by` is important
- For queries with a `group by` but no `order by`, the order was still predictable in pre-15.0:
 - ✓ Result set order = sorted order of `group by` keys
- Due to hash-based sorting in 15.0, this order is not predictable anymore
 - ✓ ...and that's a problem for some customers

Group by ordering



In pre-15.0:

```
1> select count(*), type
2> from sysobjects group by type
3> go
```

```
----- type
-----
    1 D
   55 P
   52 S
   19 U
    4 V
```

Group by ordering



In 15.0:

```
1> select count(*), type
2> from sysobjects group by type
3> go
```

```
----- type
-----
      1 V
     55 S
       3 U
     12 P
```

Group by ordering



Solution in 15.0 ESD#2:

```
1> dbcc traceon(450)
```

```
2> go
```

```
1> select count(*), type
```

```
2> from sysobjects group by type
```

```
3> go
```

```
----- type
-----
    12 P
    55 S
     3 U
     1 V
```

Things to keep in mind for ASE 15 migration



You don't need to change any SQL when upgrading to ASE 15, except when...

... you're looking at the internal names of #temp tables

Things to keep in mind for ASE 15 migration



Internal representation of #temp table names has changed in ASE 15:

```
create table #t (a int)
```

In pre-15: #t_____00000210008240896

In 15.0: #t00000150003699485

- Background: object names can now be 255 characters long

Things to keep in mind for ASE 15 migration



No functional difference, except for:

```
create table #t (a int)
```

```
create table #t__ (a int)
```

- succeeds in 15, fails in 12.5

```
create table #t12345678901 (a int)
```

```
create table #t123456789012 (a int)
```

- succeeds in 15, fails in 12.5

```
select * from #t12345678901
```

```
select * from #t1234567890123456
```

- refer to same table in 12.5, to different ones in 15

Things to keep in mind for ASE 15 migration



Determine the owning session of a #temp table:

In pre-15: #t_____00000210008240896

```
select owner_spid= substring(@name, 16, 5),  
       nestlevel=  substring(@name, 14, 2)
```

In 15.0: #t00000150003699485

```
select owner_spid= substring(right(@name,17), 3, 5),  
       nestlevel=  substring(right(@name,17), 1, 2)
```

Long identifiers



Impact of changes to sysobjects, sysindexes

- *name* field increased from sysname(30) to longsysname(255)
- Plan time to alter and test homegrown tools, scripts, stored procedures. Some things that may be useful:
 - ✓ sp_autoformat
 - ✓ convert(),
 - ✓ left()
 - ✓ Increase size of local variables defined in user created stored procedures

Things to keep in mind for ASE 15 migration



You don't need to change any SQL when upgrading to ASE 15, except when...

... directly selecting object names from system tables

Things to keep in mind for ASE 15 migration



Ad-hoc queries involving object names produce messier output:

- **sysobjects.name** is now a **varchar(255)** datatype, so it wraps 3 lines in the **isql** output

```
1> select name, id from sysobjects
```

```
2> go
```

Things to keep in mind for ASE 15 migration



```
1> select name, id from sysobjects
```

```
2> go
```

```
name
```

```
id
```

```
-----  
-----  
-----  
-----  
sysobjects
```

```
1  
sysindexes
```

```
2  
[...]
```

Things to keep in mind for ASE 15 migration



Solutions:

- `select left(name,20) id from sysobjects`

- use **sp_autoformat**:

```
sp_autoformat table_name [, @selectlist [,  
@whereclause [, @orderby [, @fmtspecifier ]]]]
```

Procedure Cache



Procedure cache is a shared resource

- Best known for holding query plans
 - ✓ Query plans for stored procedures, triggers and dynamic sql are stored in the procedure cache
 - ✓ Query plans for stored procedures and triggers are left to be re-used once their execution is complete
 - ✓ Defaults, rules and views are shared within procedure cache

Procedure Cache



...but some procedure cache is also used for other purposes:

- Sort operations
 - ✓ **create index, update statistics** on non-leading index columns, **group by, order by**, etc.

- Subquerycache
- Query optimizer
- In 15.0: 16Kb per (scrollable) cursor
- LWPs (regular SQL statements converted to a stored procedure)
 - ✓ Among others, when statement cache is enabled

- ... and various other, low-level things

Procedure Cache Usage



Why bother about procedure cache?

- If not enough procedure cache space can be made available, the 701 error is raised:

```
Msg 701, Level 17, State 3
```

```
Server 'PROD', Line 1
```

```
There is not enough procedure cache to run this  
procedure, trigger, or SQL batch. Retry later, or  
ask your SA to reconfigure SQL Server with more  
procedure cache.
```

Familiar?

- Often seen when running **update index statistics** for the first time

Procedure Cache Usage



Fact: ASE 15 will use more procedure cache than pre-15

- Query optimizer needs more memory than in pre-15
- New sorting methods use more proc.cache memory (but less tempdb space)
 - ✓ Hash-based / streaming access methods
 - ✓ Used for worktable optimization
- Hash joins
- Etc...

Procedure Cache



So for various reasons, ASE 15 will use more procedure cache than pre-15

- How much more exactly is difficult to say
- Default '**procedure cache size**' has been increased to 14 Mb

Procedure Cache



Procedure cache usage per statement

- **monSysStatement.memUsageKB**: max.amount of proc.cache used for each statement
 - ✓ ASE 15.0 & 12.5.4
 - ✓ In ASE pre-12.5.4, this value is useless
- **monProcessStatement.MemUsageKB** : memory currently allocated by a spid (same as **sysprocesses.memusage**)

Procedure Cache



Monitor procedure cache usage with `sp_monitorconfig 'all'` :

In pre-12.5.2:

Name	Num_free	Num_active	Pct_act	Max_Used	Reused
-----	-----	-----	-----	-----	-----
procedure cache size	3105	166	5.07	3232	No

In 12.5.2+:

Name	Num_free	Num_active	Pct_act	Max_Used	Num_Reuse
-----	-----	-----	-----	-----	-----
procedure cache size	3105	166	5.07	3232	8

- **Num_Reuse** has special meaning for procedure cache:
 - ✓ the number of query plans that have been removed from the cache to make space; not the number of times a 701 error occurred

Procedure Cache



Query plan size in procedure cache:

```
1> select ObjectName, CompileDate, MemUsageKB
2> from monCachedProcedures
3> go
```

ObjectName	CompileDate	MemUsageKB
sp_who	Aug 19 2005 2:03PM	42
my_proc	Aug 19 2005 3:48PM	524

(etc.)

Update statistics in ASE 15

Update Statistics in ASE 15



Statistics are more important in ASE 15 than before

- In ASE 15, statistics are used for more purposes than before
- For example, join costing is no longer based on the density (basically, 1 float number per table), but 'join histograms' are used instead
 - ✓ The histograms on the join columns are merged, and these are used instead of the density to estimate the #duplicate keys
- With the increased number of possible join types and access algorithms, having good statistics is more important than ever

Update Statistics in ASE 15



Recommendations

- Recreate all statistics after upgrading to 15
 - ✓ Consider deleting all existing stats first (**delete statistics**)
 - ✓ ... then recreate them

- Consider using larger default step count (or increasing in update stats)
 - ✓ Particularly for large tables (> 1 million rows)
 - config param '**number of histogram steps**'
 - **update statistics my_table using 400 values**
 - don't go wild here, especially not when '**histogram tuning factor**' > 1

- Use 'histogram tuning factor' or large step count if data values are skewed
 - ✓ '**histogram tuning factor**': default=20 as of 15.0.1 esd#1
 - ✓ when > 1, increases #histogram steps when duplicates are found; otherwise no effect

- Run **update index statistics** if possible
 - ✓ May need to use sampling for large tables (next slide)

Update Statistics in ASE 15



- Use **datachange()** to decide when update statistics needs to be run
 - ✓ per table, partition or column

- On partitioned tables – only run update stats on partitions with 'significant' amount of new/changed rows

- Don't forget update statistics with sampling
 - ✓ useful for large tables
 - ✓ applies to non-leading index columns and non-indexed columns only
 - ✓ **update index statistics my_table with sampling=5 [percent]**

Update Statistics in ASE 15



A classic question: When to run update statistics?

- New built-in: **datachange()** provides the user an accurate measure of the actual change in data distribution

datachange(*table_name*, *partition_name*, *column_name* **)**

- **datachange()** provides the number of inserts, updates and deletes [DML] operations on a table or partition, optionally for a specific column
- Expressed as a percentage of the total number of rows in the table or partition since the last **update statistics** execution
- Can be used to initiate **update statistics** for a column, partition or table

Automatic Update Statistics: Job Scheduler



Job Scheduler templates provided

In addition, the user can define rules at which point update statistics would need to run

In this example update statistics would run only if the data on the object authors has changed more than 10% since the last update statistics execution.

```
select @datachange = datachange('authors', null, null)
if @datachange > 10
begin
    update index statistics authors
end
go
```

Optimistic recompilation



Optimistic recompilation in ASE 15

- When an index is created on a table, all query plans referring to that table are recompiled
 - ✓ Assuming a new index is created for a good reason
 - ✓ Both for function indexes and 'normal' indexes
- In pre-15, recompilation happened only when an index was dropped, or when you do **sp_recompile**

Optimistic recompilation



NB: As of ASE 12.5.1, no `sp_recompile` is needed anymore following update statistics

- All query plans referring to the table will be recompiled automatically
- But **`sp_recompile`** doesn't hurt, so you don't have to change anything

Upgrade Considerations (3)



300 Series Diagnostic Trace Flags

- E.g. 302,310, 311,315, etc.
- Deprecated with showplan options in ASE 15
- Some don't work now – all will be no longer functional in future releases
- 302 is replaced by **set option show_ljo_costing on**
 - ✓ requires TF 3604 (otherwise output goes to console)

Query Plan Troubleshooting



Previous required trace flags:

- 310, 311, 315, 317, etc.
- Killed reams of paper and drowned DBA in too much minutiae

ASE 15 – Controllable levels of detail

- Eliminates need for trace flags
 - ✓ Except 3604 for diagnostic output (**set option show_....**)
- Optimizer diagnostics:

```
set option show {normal | long | brief | on | off }
```

Query Tree (includes estimated vs. actual LIO/PIO/rowcount):

```
set statistics plancost on
```

→ always provide this output when reporting query plan problems

Query Plan Troubleshooting



- Missing Statistics

```
set option show_missing_stats {on|brief|normal|long|off}
```

- Output (to client: requires TF 3604; otherwise goes to console)

```
NO STATS on column t1.a1
```

```
NO STATS on column t2.b1
```

```
NO STATS on density set for t1={a2, a1}
```

- Create histograms/densities for these columns

```
update statistics t1 (a1)
```

```
update statistics t2 (b1)
```

```
update statistics t1 (a2,a1)
```

Questions, Comments...



Rob Verschoor

**Senior Technology Evangelist
Sybase Engineering
robv@sybase.com**



SYBASE®