UPGRADING TO ASE 15
An in-depth view of the (un)expected

Rob Verschoor
Senior Technology Evangelist
Sybase, Inc.
WHAT WE’LL BE TALKING ABOUT

• Upgrading to ASE 15
  – Why upgrade?
  – How to perform the upgrade
  – ASE 15 QP aspects & migration strategies
  – ASE configuration impact
  – Potential changes to existing SQL code
  – Update statistics in ASE 15
WHY UPGRADE 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, ...)

SYBASE®
ASE 15 QUERY PROCESSING

• ASE 15: released end of 2005
  – Latest version as of March 2009: 15.0.3 esd#1
  – EOL for 12.5 is 31-Dec-2009

• Recommendation:
  – Please get the latest ASE 15 patch if you run into any issues!
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 as 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
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
UPGRADING – RESOURCES

• Please visit: www.sybase.com/ase15migration
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 **preupgrade** tool
  – Fix any issues reported
  – Run **preupgrade** 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 \(\rightarrow\) 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**...)
    ▪ Only if you’re unhappy with performance... and make a backup first
  – ... 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 TechSupport 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

![Bar Chart](attachment:image.png)

- **290% Gain**
- (smaller is better)
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 TechSupport is your friend here (aren’t they always?)
  – SySAM 2.0 also allows you to set up a redundant license servers for HA systems
ASE 15 QUERY PROCESSING & MIGRATION STRATEGIES
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

• You cannot switch back to the 12.5 optimizer+exec engine!
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:

  • **allrows_oltp** - uses a limited number of optimization criteria to find a good query plan aimed at OLTP
    – nested loop join, no parallelism

  • **allrows_mix** - (default) generates plans for mixed workloads
    – basically, **allrows_oltp** + merge joins, parallelism

  • **allrows_dss** - generates optimal plans for high-complexity DSS
    – basically, **allrows_mix** + hash joins
QUERY PROCESSING ENHANCEMENTS

• Performance improvements for DSS / complex queries / aggregates / star schemas:
  – Use allrows_dss or allrows_mix

• Performance improvements for OLTP:
  – Use allrows_oltp
  – Use semantic 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, usually not to business applications
    – Possible exception: specific usage of cursors
OPTIMIZATION GOALS IN 15.0

• Optimization goal can be set at server, session and query level:
  – Server:
    `sp_configure "optimization goal", 0, "allrows_dss"
  – Session (can be done in a login trigger!):
    `set plan optgoal allrows_dss`
  – Query:
    `select * from A order by A.a`  
    `plan "(use optgoal allrows_dss)"`
OPTIMIZATION GOALS IN 15.0

- Objective with optgoals should be: Keep It Simple
  - Try using as little fine-tuning as possible
    - Ideal: just one server-wide setting and no lower-level overrides
      - ...though 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
  - Individual queries might benefit from a different plan than generated by the current optgoal
    - Quickly becomes complex and time-consuming
    - QPTune, a new tool, will assist here
OPTIMIZATION GOALS IN 15.0

• To experiment 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
    ▪ Use QPTune to determine the best query plan for particular query
      – Expected in 15.0.3 ESD#1
ASE 15 MIGRATION STRATEGIES

- Objective:
  migrate to ASE 15 without using 15-specific QP features
  - 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
ASE 15 MIGRATION STRATEGIES

• Objective:
migrate to ASE 15, and start using 15-specific QP features
  – Run your application under each optgoal
    ▪ Try allrows_oltp and allrows_mix and allrows_dss as server-wide optgoals, one by one...
    ▪ ... and determine whe your application runs best
      – This is easier said than done!
    ▪ New tool: QPTune (expected 15.0.3 esd#1) will make it easier to interpret the results
      – Some queries may run better under allrows_dss, some run better under allrows_oltp.... How do you pick these?
ASE 15 MIGRATION STRATEGIES

• QPTune (expected 15.0.3 ESD#1)
  – Based on capture of abstract plans and query metrics...
  – ... and abstract plan load ('AP association')
  – First, QPTune lets you determine the best plan for a particular query, based on multiple executions of the application
  – Next, QPTune lets you specify that best plan for that particular query, ‘forcing’ that statement always to use that plan
    ▪ No need to understand or write abstract plans!
    ▪ No need to change SQL code
    ▪ No need to micro-manage optimization goals
      – ...since an abstract plan overrides the optimizer
  – QPTune will be a client application
ASE 15 MIGRATION STRATEGIES

• Applies in all cases:
  – Use `set option show_missing_stats on`, and consider following up by creating additional statistics or indexes
    - In 15.0.3 esd#1 'capture missing statistics' collects this information in the `sysstatistics` table
  – 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 allowed 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 \rightarrow www.sypron.nl/exectab.html)
  – If things get faster \rightarrow good!
  – If things get slower \rightarrow try to find out which ones
    ▪ Use MDA tables, run those queries with a different optgoal
    ▪ Use QPTune (expected 15.0.3 esd#1)
'COMPATIBILITY MODE' IN 15.0.3 ESD#1

- ASE 15.0.3 esd#1 supports 'compatibility mode'
  - Using optimizer/execution logic similar to 12.5
  - When enabled, each optimized SQL statement can go:
    1. through the fully 12.5-compatible mode
    2. ... or else through the partial 12.5-compatible mode
    3. ... or else through the ASE 15 mode

  - Under 1. and 2., ASE 15 optimization aspects do not apply
    - Under 1., 12.5-specific commands like `set sort_merge on` now apply again
  
  - When ASE-15-specific features are used, the ASE 15 mode will always be chosen

  - server level: config param 'enable compatibility mode' → set to 1
  - session level: `set compatibility_mode on/off`
STATEMENT CACHE IN 15.0.1

• Statement cache
  – Introduced on 12.5.2
  – Not broadly useful until ‘literal autoparam’ was added in 15.0.1

```sql
select * from my_table where k = 123
select * from my_table where k = 124
```

➤ without literal autoparam: both query plans are cached separately
  (a query must be **exactly** identical for the cached plan to be re-used)

➤ with literal autoparam: the same cached query plans are cached separately
  Literal autoparam replaces the constants with placeholders for the cached plan:
  ```sql
  select * from my_table where k = @@V0_INT
  ```

**Recommendation:**
– When running OLTP workloads, enable the statement cache
  ▪ and **always** enable literal autoparam!
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 TechSupport)
      – ... or you have time left
ASE 15 QUERY PROCESSING

• Advanced optimizer tuning options
  – Resource granularity settings
  – Optimization timeout limits
  – Repartition degrees
  – Same advice: don’t go there unless advised by TechSupport
‘DEFERRED COMPILATION’ IN 15.0.2

• Classic ASE optimizer problem
  – Queries referring to local @variables or #temp tables may get a sub-optimal plan
    
    ```sql
    create proc p1 as
    declare @a int
    select @a = max(col) from your_tab
    select * from my_tab
    where col = @a
    go
    ```

  – Optimizer uses magic numbers to cost the predicate 'col = @a'
    • Unlikely to be accurate
‘DEFERRED COMPILATION’ IN 15.0.2

• Classic ASE optimizer problem
  – Queries referring to local @variables or #temp tables may get a sub-optimal plan

```sql
create proc p2 as
  select * into #t from my_tab

  select * from #t, your_tab
  where col1 = col2

go
```

– Optimizer assumes #t has 100 rows, 10 pages
  ▪ Unlikely to be accurate
‘DEFERRED COMPILATION’ IN 15.0.2

• Solution: deferred compilation
  – Queries referring to local @variables or #temp tables are compiled when they are first executed
    ▪ Value of local @variables is known
    ▪ Size of #temp tables is known
  – Subsequent executions of such a statement will re-use the query plan generated at the first execution
    ▪ Unless the proc is executed with recompile
  – Deferred compilation was introduced in 15.0.2
    ▪ Enabled by default, switch off with traceflag 7730 in pre-15.0.3 esd#1
    ▪ In 15.0.3 esd#1, governed by config parameter 'procedure deferred compilation'
    ▪ Currently only applies to stored procs
    ▪ Currently does not apply to select-into statements
  – We’ve seen great results from deferred compilation!
ASE CONFIGURATION IMPACT WHEN MIGRATING TO ASE 15
THINGS TO KEEP IN MIND FOR ASE 15 MIGRATION

- New system table: `syspartitions`
  - Pre-15 `syspartitions` is renamed `sysslices` 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:

    ```
    sysindexes   syspartitions
    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_countmetadatad 'number of open partitions'` for an initial estimate
  - **sysindexes** columns remain, but values will be 0 or unreliable
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.
  – Subquery cache
  – 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
  – Since 15.0.2 IR, procedure cache consumption has been reduced
    ▪ Especially cases of excessive usage
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:
  
<table>
<thead>
<tr>
<th>Name</th>
<th>Num_free</th>
<th>Num_active</th>
<th>Pct_act</th>
<th>Max_Used</th>
<th>Reused</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure cache size</td>
<td>3105</td>
<td>166</td>
<td>5.07</td>
<td>3232</td>
<td>No</td>
</tr>
</tbody>
</table>

- In 12.5.2+:
  
<table>
<thead>
<tr>
<th>Name</th>
<th>Num_free</th>
<th>Num_active</th>
<th>Pct_act</th>
<th>Max_Used</th>
<th>Num_Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure cache size</td>
<td>3105</td>
<td>166</td>
<td>5.07</td>
<td>3232</td>
<td>8</td>
</tr>
</tbody>
</table>

  – **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.)
```
POTENTIAL SQL CODE CHANGES WHEN MIGRATING TO ASE 15
OAM PAGE BUILT-IN FUNCTIONS

- You don’t need to change any SQL when upgrading to ASE 15, except when...
- ... calling the storage space builtins:
  - rowcnt()
  - data_pgs()
  - ptn_data_pgs()
  - reserved_pgs()
  - used_pgs()
  - ... and possibly a few other, specific cases
OAM PAGE BUILT-IN FUNCTIONS

• 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]])
OAM PAGE BUILT-IN FUNCTIONS

• 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.
SYSDEVICES / SYSUSAGES

• You don’t need to change any SQL when upgrading to ASE 15, except when...
• ...doing direct queries against:
  – sysdevices
  – sysusages
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`
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>
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 (see next slides)
    ▪ Can use code from `sp_helpdevice` as a model for new custom stored procedures
SP_HELPDEVICE

1> sp_helpdevice dev1
2> go

<table>
<thead>
<tr>
<th>device_name</th>
<th>physical_name</th>
<th>description</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[...]</td>
<td>physical disk, 900.00 MB, Free: 235.00 MB</td>
<td>16386</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

(1 row affected)
1> sp_helpdevice dev1
2> go

device_name    physical_name    description                 status

<table>
<thead>
<tr>
<th>cntrltype</th>
<th>vdevno</th>
<th>vpn_low</th>
<th>vpn_high</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1 row affected)

dbname    size    allocated        vstart      lstart

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>-----------</td>
<td>------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>z1</td>
<td>350.00 MB</td>
<td>Sep 20 2005 8:01:56:346PM</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tempdb</td>
<td>4.00 MB</td>
<td>Sep 27 2005 2:23:56:430PM</td>
<td>307200</td>
<td>2048</td>
</tr>
<tr>
<td>demo3</td>
<td>250.00 MB</td>
<td>Jan 5 2006 6:53:26:643PM</td>
<td>179200</td>
<td>0</td>
</tr>
<tr>
<td>tempdb</td>
<td>6.00 MB</td>
<td>Apr 4 2006 7:34:55:430PM</td>
<td>309248</td>
<td>4096</td>
</tr>
<tr>
<td>demo</td>
<td>5.00 MB</td>
<td>Jun 7 2006 1:02:09:543PM</td>
<td>312320</td>
<td>0</td>
</tr>
<tr>
<td>demo2</td>
<td>50.00 MB</td>
<td>Jun 7 2006 1:02:10:543PM</td>
<td>314880</td>
<td>0</td>
</tr>
</tbody>
</table>
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
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 = $2^{24}*2^{Kpg} = 32GB$
      – 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)
### VLDB ENHANCEMENTS IN 15.X

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Old (12.5.x) Limit</th>
<th>New (15.0) Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of devices</td>
<td>256</td>
<td>2,147,483,648 (2^31)</td>
</tr>
<tr>
<td>Maximum device size</td>
<td>32 Gb</td>
<td>4 Tb</td>
</tr>
<tr>
<td>Maximum database size (2K / 4K / 8K / 16K)</td>
<td>4 Tb / ~8 Tb / ~8 Tb / ~8 Tb</td>
<td>4 Tb / 8 Tb / 16 Tb / 32 Tb</td>
</tr>
</tbody>
</table>

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.
EXPANDED STORAGE VOLUME IN ASE 15

• 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 $\Rightarrow$ 8,589,934,592 TB
  – Theoretical DB storage
    ▪ 32,767 DB’s * 32TB = 1 EB (exabyte) = 1,048,544 TB
OBJECT ID IN PAGE HEADER

- 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))
To determine the object for a given partition ID:

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

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

```sql
select object_name(partition_object_id(ptnid))
```
GROUP BY ORDERING

• You don’t need to change any SQL when upgrading to ASE 15, except when...
• ... using a non-ANSI approach towards order-by
GROUP BY ORDERING

- 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

<table>
<thead>
<tr>
<th>type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>55</td>
</tr>
<tr>
<td>P</td>
<td>52</td>
</tr>
<tr>
<td>S</td>
<td>19</td>
</tr>
<tr>
<td>U</td>
<td>4</td>
</tr>
</tbody>
</table>
```
GROUP BY ORDERING

• In 15.0:

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

<table>
<thead>
<tr>
<th>type</th>
<th>-------</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V</td>
</tr>
<tr>
<td>55</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
</tr>
<tr>
<td>12</td>
<td>P</td>
</tr>
</tbody>
</table>
```
GROUP BY ORDERING

• Solution in 15.0 ESD#2:

```sql
1> dbcc traceon(450)
2> go
1> select count(*), type
2> from sysobjects group by type
3> go

<table>
<thead>
<tr>
<th>type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>55</td>
</tr>
<tr>
<td>U</td>
<td>3</td>
</tr>
<tr>
<td>V</td>
<td>1</td>
</tr>
</tbody>
</table>
```
INTERNAL #TEMP TABLE NAMES

• 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
INTERNAL #TEMP TABLE NAMES

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

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

In pre-15:
```
#t___________00000210008240896
```

In 15.0:
```
#t00000150003699485
```

– Background: object names can now be 255 characters long
INTERNAL #TEMP TABLE NAMES

• No functional difference, except for:

```sql
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
```
• 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)
```
USING PRE-15-STYLE PARTITIONED TABLES

• 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 (see next slide)
  – NB: actually, unpartitioned tables don’t really exist anymore in 15
    ▪ Each table has at least 1 roundrobin partition when created without further partitioning
UNPARTITIONING PRE-15-STYLE PARTITIONED TABLES

• ASE supports partitioning since version 11.0
  – purpose is to reduce last-page lock contention for concurrent inserts

• In ASE 15, this is known as 'round-robin' partitioning following new partition syntax

• However, the pre-15 syntax is still supported...
  ... with one difference:

    alter table t unpartition

    – this statement always works in 12.5 for a partitioned table...
    – ...but fails in ASE 15 if the table has any indexes

• In ASE 15, update any scripts using alter table...unpartition, and drop indexes first
  – if this occurs at all, likely only for defragmenting/recreating partitioned tables
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
LONG IDENTIFIERS

• You don’t need to change any SQL when upgrading to ASE 15, except when...
• ... directly selecting object names from system tables
LONG IDENTIFIERS

• 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
```
LONG IDENTIFIERS

1> select name, id from sysobjects
2> go

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sysobjects</td>
<td>1</td>
</tr>
<tr>
<td>sysindexes</td>
<td>2</td>
</tr>
</tbody>
</table>

[...]
LONG IDENTIFIERS

• Solutions:
  – `select left(name,20) id from sysobjects`
  – use `sp_autoformat`:

    `sp_autoformat table_name [, @selectlist [, @whereclause [, @orderby [, @fmtspecifier ]]]]]`
USING CURSORS WHILE MODIFYING TABLES

• You don’t need to change any SQL when upgrading to ASE 15, except when...
• ... you’re using cursors AND you’re modifying rows in the table accessed by the cursor
USING CURSORS WHILE MODIFYING TABLES

• When using a cursor with a worktable in 12.5, the entire result set is cached (in another worktable) before FETCHing begins
  – If rows in the underlying table are deleted/modified while the cursor is open, this will not affect the rows being FETCHed
    ▪ In ASE 15 this is called an insensitive cursor

• Worktables may be used for things like distinct
  – ASE 15 may no longer need a worktable here due to better algorithms like hash-based distinct
    ▪ ...so the cursor result set is not fully cached first
  – Now, if rows in the underlying table are deleted/modified while the cursor is open, this may affect the rows being FETCHed
    ▪ In ASE 15 this is called a semi_sensitive cursor
USING CURSORS WHILE MODIFYING TABLES

Session 1

declare curs1 cursor
  for select distinct c1, c2
  from t1
-- assuming > 1 row found...
open curs1
fetch curs1

fetch curs1

Session 2

delete from t1

• In 12.5, the last fetch will return 1 row
• In 15, the last fetch may or may not return a row
• In other words, a cursor-based application in ASE 15 may behave differently than it did in 12.5
  – Applies only under a series of rather specific conditions!
  – Solution 1:
    ▪ Change cursor declarations to include explicit `insensitive` keyword in ASE 15
      – default in ASE 15 is `semi_sensitive`

    ```sql
    declare my_curs insensitive cursor
    for select a,b,c from t
    ```
  – Solution 2:
    ▪ Use traceflag 457 to make this category of cursors insensitive
      – Requires 15.0.2 ESD#6
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
  – Run **update index statistics** after upgrading to 15
  – 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
    ▪ when HTF set to default of 20, and > 400 steps requested by update stats, max. 400 steps will be generated to avoid step count explosion
  – 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

• If still unhappy with performance...
  – Check if old statistics exist
  – If so, consider recreating all statistics after upgrading to 15
    ▪ First, back up all existing stats with optdiag
    ▪ Delete all existing stats (delete statistics)
    ▪ ... then recreate them
      – Important: don't loose the original step count (get from optdiag output)
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.

```sql
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_lio_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 detail
• 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)