SAP Sybase ASE in 2022

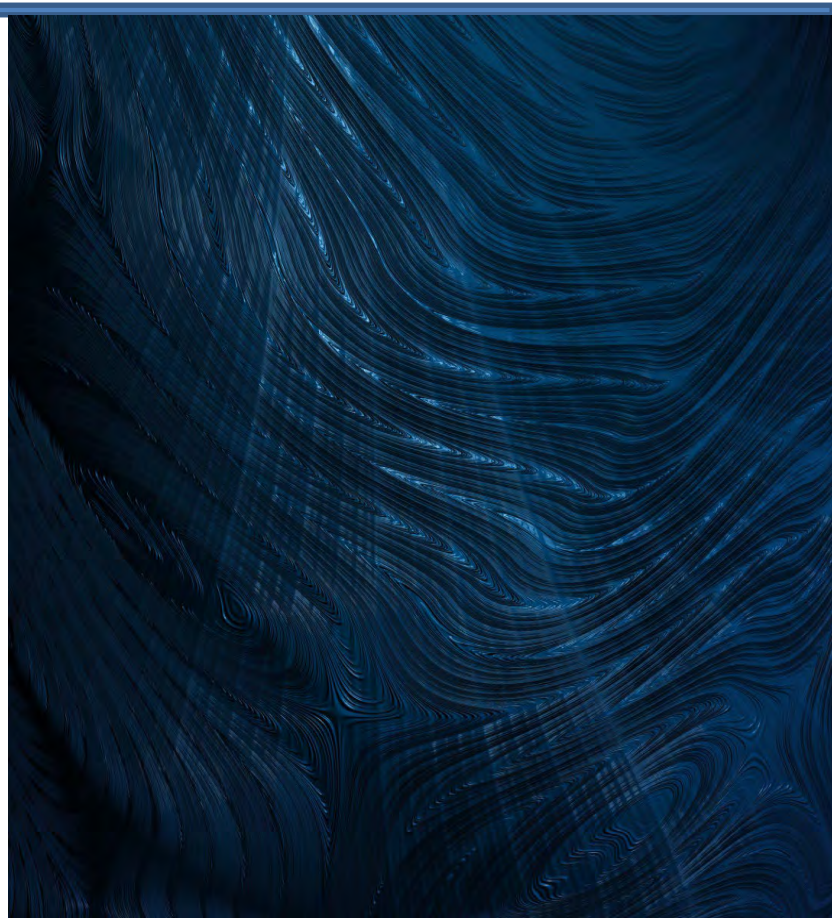
Are you still running ASE like it was 2012?

Joe Woodhouse
17 February 2022

PRIMA DONNA CONSULTING

Contents

- About the author & this presentation
- Some IT philosophy, briefly
- Part one: your feature delete list
- Part two: your feature update list
- Part three: your feature insert list
- Join / renew ISUG with discount code 2022joinisug by 28 February 2022



About the author

- Sybase Australia 1996 – 2003
- Database engineer via Prima Donna Consulting for 19+ years
- Works exclusively with ASE, IQ, and Replication Server for 26+ years
- Based in London, UK and Melbourne, Australia
- International Sybase User Group Board of Directors since 2010
- UK Sybase User Group Board of Directors since 2019
- Not a lawyer – no charge for emails!

Why this presentation?

- Most sites using ASE seem slow to adopt new features
  - … including "new" features that aren't really all that new now
- My experience is that most sites patch the binaries and little more
- It is prudent to adopt new features slowly…
  - … but waiting 10+ years means we maintain much toil
- SAP don't really maintain a clear deprecation list… and it should

First, some IT philosophy (1 of 3)

- Toil
  - Not the same as overhead – some chores must always be done
  - Not the same as "I don't wanna"
- Google SRE defines it well
  - "Toil is [...] manual, repetitive, automatable, tactical, devoid of enduring value, and scales linearly as a service grows".

First, some IT philosophy (2 of 3)

- Technical debt
  - The cost of rework caused by choosing an easier/faster solution instead of the better but slower solution
  - Not always a bad thing
    - » Debt enables what there were not enough resources to achieve
  - Like all debt it must be repaid
  - Like all debt it accrues interest
    - » The cost to repay it increases with time

First, some IT philosophy (3 of 3)

- IT systems entropy
- Changing & extending IT systems causes them to rot
  - They require care to maintain design principles and patterns
- IT systems rot even if left untouched
  - The IT landscape changes
    - » Systems are upgraded, replaced or fall out of support
  - The business and its requirements change
- Like physical entropy: you cannot win and you cannot break even

Approach for this presentation (1 of 2)

- This isn't a "new features" presentation
  - It's not enough for a feature to be new
- To appear here it must:
  - be available but seldom used, and must also (at least one)
    - reduce toil, or
    - repay technical debt, or
    - reduce systems entropy
- Often means replacing an anti-pattern with a good design pattern

Approach for this presentation (2 of 2)

- Part one: your delete list
  - Deprecated features, it's time to stop doing these things
- Part two: your update list
  - You already do these things, it's time to do them differently
- Part three: your insert list
  - If you aren't doing these things, it's time to start

Part One
ASE Feature Delete List
aka ASE Deprecated Features

ASE deprecation list – why (1 of 1)

- All of this is documented… somewhere…
- Often features evolve from one part of ASE to another
  - Very common: dbcc ⟹ trace flag ⟹ sp_configure
- Some features remain for backwards compatibility… for a while…
  - These can disappear with little notice
- Some features change their behaviour
  - e.g. trace flags whose behaviour is reversed in an ASE patch

Optimizer sins (1 of 3)

- Forceplan / forceindex
- These go back to at least SQL Server 11.0 (1997)
- Not just "hints", these give the optimizer no choice at all
- They were perhaps once necessary for specific bugs
  - Classic technical debt & code entropy!
  - Once in the code are they ever reviewed?!
- Too hard to remove from code? (Can you even tell with COTS??)
  - Try trace flags –T15307 and –T15308

Optimizer sins (2 of 3)

- ASE compatibility mode (sp_configure)
- Introduced in ASE 15.0.3 (2009)
- Makes ASE optimiser behave as it did in ASE 12.5.4 (2006)
  - Seen in ASE 16 PROD in 2020!
- Perhaps this made sense in 2009 but not today
- Eliminate this anywhere you find it

Optimizer sins (3 of 3)

- Optimizer level (sp_configure)
  - This is compatibility mode in a different uniform
- Added in ASE 15.0.3 ESD #2 (still 2009)
- "ase_default" is still the default setting in 2022
  - It means run like ASE 15.0.3 ESD #1 (2009)
- Anti-pattern: good intentions but did more harm than good
- Seen in the wild even today
- Set this to "ase_current" anywhere you find it
  - This still doesn't switch every optimizer fix on!

Kernel mode (1 of 2)

- ASE 15.7 (2011) introduced the threaded kernel and made it default
- Some legitimate issues through 15.7.x and persisting into 16.0.x
  - ASE 15.7 ESD #1 (2012), sp_configure "network polling mode"
  - ASE 15.7 SP100 (2013), sp_configure "solaris async i/o mode"

    + Oracle Solaris fix for BugID 16054425
  - ASE 16.0 SP03 PL02 (2017), sp_configure "async poll timeout"

    + trace flag 7866

Kernel mode (2 of 2)

- Also a lot of technical debt because sites changed *just* the kernel mode
- Some OS kernel tuning was required
  - Mostly per-process limits as now only one process
- Shouldn't keep number of engines the same
  - … could write a presentation just for tuning the threaded kernel
- The point is that the old process kernel is officially deprecated
  - It might disappear in the next patch
  - If you're still using it today, it's time to test and plan the switch to threaded kernel

Trace flags (1 of 7)

- These are inherently unsafe, even SAP says so
- Not guaranteed to still exist or to behave the same after any patch
- Must be reviewed after every patch
  - Great example of system entropy/rot

Trace flags (2 of 7)

- -T326 obsolete as of ASE 11.5 (replaced by sp_modifystats)
- -T291 obsolete as of ASE 11.9.2 (no longer needed)
- -T107 obsolete as of ASE 12.0 (no longer needed)
- -T302/303/310/311/317/319/321/334/384 obsolete as of ASE 12.5
  - unless compatibility mode enabled… but you won't do that
- -T303/320/324/333 obsolete as of ASE 12.5 even with compatibility
  - all 3xx replaced by various statistics/showplan enhancements

Trace flags (3 of 7)

- -T8399 obsolete as of ASE 12.5.0.2 (no longer needed)
- -T4044 obsolete as of ASE 12.5.0.3 (replaced by dataserver -u)
- -T701 obsolete as of ASE 15.0.1 (no longer needed)
- -T109 obsolete as of ASE 15.0.2 (reverse behaviour now in –T114)
- -T7841 obsolete as of ASE 15.0.2 ESD 5 (no longer needed)
- -T12314 obsolete as of ASE 15.7 ESD 4 (no longer needed)

Trace flags (4 of 7)

- -T299 obsolete as of ASE 16.0
  - replaced by sp_configure "optimize temp table resolution"
- -T753 obsolete as of ASE 16.0
  - replaced by sp_configure "enable large chunk elc"
- -T1102 obsolete as of ASE 16.0
  - replaced by sp_altermessage 1105, "with_log", true

Trace flags (5 of 7)

- You probably think this is all old news and doesn't affect you today…
- -T893 obsolete as of ASE 16.0 SP02 PL09 (replaced by -T891/894)
  - April 2020
- -T3945 behaviour now obsolete (on by default) in ASE 16.0 SP03 PL08
  - -T3996 reverses this (i.e. reverts to previous default)
  - March 2020
- Trace flag behaviour may change, disappear, or reverse at any time

Trace flags (6 of 7)

- There is a special hell reserved for procedure cache trace flags
- So much folklore and superstition online!
- Most problems caused by over- or mis-configuration
  - … and by not fully understanding procedure cache
  - To be fair this is one of the most complicated parts of ASE
- Any procedure cache problems must start with "right-sizing" ASE configuration (more on this later)
- ASE 16.0.x proc cache spinlock contention tuning process on next slide

Trace flags (7 of 7)

- Don't use -T753 (deprecated), –T757, or dbcc proc_buf(free_unused)
- Start with sp_configure "large allocation auto tune" and "enable large chunk elc" (replaces –T753) enabled
- If MDA shows issue with procedure cache, then boot ASE with -T758
- If problem persists, boot ASE *also* with –T757
- If problem persists, boot ASE with *only* -T757 (remove –T758)
- If problem persists, remove both –T757 and –T758, *and* disable "large allocation auto tune" and "enable large chunk elc"

Time to stop using these too (1 of 3)

- sp_dboption "async log service"
  - Use sp_configure "user log cache queue size" (more on this soon)
  - Use sp_dboption "delayed commit" only for staging/scratch dbs
- dbcc tune(des_bind [...])
  - Use sp_configure "scavenge temp objects", 0 (default = 1)
- create proc ... with recompile
  - Don't even use exec proc ... with recompile!
  - Use select ... with recompile only when needed per statement

Time to stop using these too (2 of 3)

- sp_configure "i/o polling process count"
  - Use sp_configure "number of disk tasks" (… carefully)
- sp_configure "runnable process search count"
  - Use thread pool idle timeout (default = 100µs)
  - Maybe increase to 150/200/250 for apps
  - Maybe decrease to 50 for DBA and batch
  - … you are using multiple user defined thread pools, right?
  - … you are using threaded kernel, *right??*

Time to stop using these too (3 of 3)

- sp_configure "user log cache size"
  - Use sp_configure "user log cache queue size"
    - » Doesn't need configuration, just needs to be enabled
- update statistics ... with consumers [...]
- update statistics ... with sampling [...]
  - Use hashing instead of either of these
  - More on this later

Part Two
ASE Feature Upgrade List
aka "you're doing this wrong"

Right-sizing (1 of 12)

- Almost all ASE tuning guides and experts suggest *more* and *bigger*
- Certainly if you see 100% ASE CPU busy or 3000+ IOPS you need more and bigger hardware
- But over-configuring some resources is actively harmful
  - Too many engines (common problem!)
  - Too much procedure cache (common problem!)
  - Even too much data cache (uncommon but happens)
  - "number of user connections" (common problem!)
  - "number of sort buffers" (common problem!)

Right-sizing (2 of 12)

- Too many engines
  - engine local cache (ELC) reserved per engine configured, even if unused/offline
  - more engines will not magically improve every workload
  - it slows some workloads down
  - generally due to contention
    - » spinlock contention on shared internals
    - » regular lock contention & deadlocks on data/index

Right-sizing (3 of 12)

- Tuning the number of engines
- N engines in process kernel = N-2 engines in threaded kernel
  - This is because threaded kernel has dedicated threads
    - "number of disk tasks" (default = 1)
    - "number of network tasks" (default = 1)
  - If you increase these (sometimes useful but not often)
    - … decrease # of threads used for all user tasks

Right-sizing (4 of 12)

- Tuning the number of engines
- Goal is to see ASE total engine busy around 80%
- If you have engine busy < 80%
  - you should consider reducing your number of engines
- "Too many cooks in the kitchen"
  - What you lose in response time you may more than gain in easing contention
    - » Especially if you already have contention issues

Right-sizing (5 of 12)

- Tuning procedure cache
- Could make a whole presentation just on this…
- This is a tricky one because procedure requirements *do* increase
  - ASE 15.7 if any of a number of features are used
  - ASE 15.7 ⟹ ASE 16.0 out of the box
  - ASE 16.0.x if any of a number of new features are used
- Alas, many of us have been bitten and say "I want to be sure I never run out of procedure cache again"

Right-sizing (6 of 12)

- Costs of too much procedure cache
  - More contention!
  - Less memory for other things that might need it more
- There are many claimed fixes for procedure cache spinlock contention
  - Most of these make things worse
  - Correct ASE 16.0 approach already discussed earlier

Right-sizing (7 of 12)

- Procedure cache sizing is complicated
  - "procedure cache size" is the one we all think of
  - "statement cache size"
    - » Actually increases procedure cache by this amount
    - » Reserves this amount only for statement cache
- Procedure cache is divided up between engines
  - Each engine gets a dedicated local amount
    - » "engine local cache percent" (default = 50)
  - Remainder is shared between all engines

Right-sizing (8 of 12)

- Procedure cache consumption is complicated
  - all queries (all query plans are in proc cache)
  - stored procedures & triggers
  - fully prepared statements from client
  - auditing!
    - » audit trail for anything in proc cache is also in proc cache

Right-sizing (9 of 12)

- Remember old training courses which try to calculate query plan size?
  - ASE 16.0.x easily uses 250-350Kb *per plan*
    - » ... but we consider more than one plan during optimization
  - ASE 16.0.x easily uses 2-5Mb *per query*
    - » ... but proc cache is not re-entrant (shared)
  - ASE 16.0.x easily uses 2-5Mb *per concurrent user per query*
- It's even worse than all that because of ELC (default 50%)
  - If you calculate 4Gb proc cache, that's really 8Gb

Right-sizing (10 of 12)

- So what's the right size???
- Easier to just "make it huge", right?
- Just as with number of engines, bigger = more contention
- I don't actually have a number for you other than to caution you
  - Some authoritative suggestions say 16+Gb is too much
- Proc cache spinlock contention might be as simple as having too much

Right-sizing (11 of 12)

- Data cache is probably the safest to oversize
  - Very seldom is it a mistake to have more
- But… consider the impact downstream when increasing data cache
  - More pages in cache = fewer physical I/Os
    - » = fewer yields of CPU
    - » = increase in CPU utilisation
    - » = increase in spinlock contention elsewhere
    - » = increased requirements for procedure cache

Right-sizing (12 of 12)

- "number of sort buffers" is routinely over-sized
  - 20 years ago this was a great tuning measure
  - Today anything > 10,000 is probably causing trouble
    - » Massive blowout of procedure cache
    - » Especially for update stats & reorg
- "number of user connections"
  - Memory cost is paid per *configured* user
  - Even if they never connect

Parallel processing (1 of 4)

- update statistics … with consumers = N
- create index … with consumers = N
- Both will use 2N+1 worker processes *per job*
  - Need sp_configure "number of worker processes" = 2N+1
    - » Just to run this one job, and no other parallel must be running
  - Need sp_configure "max utility parallel degree" = 2N+1
  - Also look at sp_configure "max repartition degree"
    - » Generally want this same as "max parallel degree"

Parallel processing (2 of 4)

- sp_configure "max parallel degree"
  - <= "number of engines at startup"
  - *NEVER* more than number of CPU cores
    - » ASE usually smart enough to prevent this
- sp_configure "max scan parallel degree"
  - Probably should never be > 4 regardless of engines
  - This is the limit of parallelism on unpartitioned tables
    - » i.e. without partitions you won't ever achieve much parallel

Parallel processing (3 of 4)

- sp_configure "memory per worker process"
  - Always under-configured, you want 65,536+
- sp_configure "min pages for parallel scan"
  - Always under-configured, you want 500,000+
- sp_configure "number of worker processes"
  - Often over-configured
  - Sensible number is 3-4 * number of engines

Parallel processing (4 of 4)

- sp_configure "max query parallel degree" obsolete
  - Only used in compatibility mode
  - … and I know you don't do that any more
- Most ASE 16.0+ improvements to parallel require "ase_current"
  - … and I know you now set that everywhere
- Moral of the story
  - ASE parallel is mostly for partitioned tables, in big hardware, using opt_goal=allrows_dss

Update statistics (1 of 14)

- Could make an entire presentation just on this…
- update statistics … with consumers […]
- update statistics … with sampling […]
  - Sampling can't be combined with parallel consumers
  - Use hashing instead of either of these
    - » *Much* faster and little effect on quality of stats

Update statistics (2 of 14)

- Stop running update statistics in PROD
  - Dump PROD, load into DR/UAT (this is good practice anyway)
  - Run update statistics in DR/UAT
  - optdiag stats out of DR/UAT, into PROD
- Good practice to dump stats …
  - … with every dump database
  - … *before* every update statistics
- Good practice to *delete* stats before every update stats
  - *DUMP STATS FIRST*

Update statistics (3 of 14)

- Run only when & where needed
- datachange() function = change since last update stats (any kind)
  - If zero or very low, skip
  - Interpret according to row count
  - datachange=5000 is fine for 1B+ rows but concerning for 1000

Update statistics (4 of 14)

- Run the right kind of update stats, on the right attributes
- Balance the goals
  - Yes we want to reduce total I/O…
  - … while producing stats on all the columns we need…
  - … without wasting effort for columns we don't need…
  - … but also reducing the total number of scans…
    - » Implications for SH table locks (if still running in PROD)
    - » More I/Os means locks are held longer

Update statistics (5 of 14)

- Here is an example table T

```
create table T (a bigint, b datetime, c datetime, d int, e varchar(50))
create index i1 on T(a)
create index i2 on T(a, b, c)
create index i3 on T(a, d)
create index i4 on T(b, c, d)
```

Update statistics (6 of 14)

Command(s)	Number of scans	Stats on a	Stats on b	Stats on c	Stats on d	Stats on e
update statistics T	1	Y	Y	N	N	N
update index statistics T	9 + 5 sorts	Y	Y	Y	Y	N

- Wait, what now?
- Nine scans?! Five sorts in worktables?!

Update statistics (7 of 14)

- It's right there in the manuals:

  The update index statistics command generates a series of update statistics operations that use the same locking, scanning, and sorting as the equivalent index-level and column-level command. For example, if the salesdetail table has a nonclustered index named sales_det_ix on salesdetail (stor_id, ord_num, title_id), the `update index statistics salesdetail` command performs these update statistics operations:

  ```
  update statistics salesdetail sales_det_ix
  update statistics salesdetail (ord_num)
  update statistics salesdetail (title_id)
  ```

- One scan per column per index
  - Columns that aren't leading columns of indexes require a sort and worktable

Update statistics (8 of 14)

Command(s)	Number of scans	Stats on a	Stats on b	Stats on c	Stats on d	Stats on e
update statistics T	1	Y	Y	N	N	N
update index statistics T	9 + 5 sorts	Y	Y	Y	Y	N
update all statistics T	5 + 3 sorts	Y	Y	Y	Y	Y

- Wait, what now?
- Only five scans, and only three sorts in worktables?!

Update statistics (9 of 14)

- It's right there in the manuals:

  Histogram statistics are created on each column, either through an index scan of a leading column, a projection of the column into a work table followed by a sort, or by using hashing to concurrently generate histograms on several columns for respective scans.

- One scan per column
- If column is not the leading column of an index
  - Copy to worktable and sort the worktable

Update statistics (10 of 14)

Command(s)	Number of scans	Stats on a	Stats on b	Stats on c	Stats on d	Stats on e
update statistics T	1	Y	Y	N	N	N
update index statistics T	9 + 5 sorts	Y	Y	Y	Y	N
update all statistics T	5 + 3 sorts	Y	Y	Y	Y	Y
update all statistics T ... with hashing	1	Y	Y	Y	Y	Y

- Wait, what now?
- Only one scan?! No sorting?!

Update statistics (11 of 14)

- It's right there in the manuals:

  When you run update all statistics [ASE] performs either a data scan followed by a sort or performs a data scan that uses hash-based statistics gathering.

  The advantage of hash-based statistics is that one data scan can collect histograms on all columns, whereas sort-based statistics uses a separate scan for each column.

- One scan per table
  - Bonus stats per column
  - Note that update index status does *not* have this optimization!

Update statistics (12 of 14)

- Update all stats … with hashing can outperform update index stats!
- Too good to be true?
  - Hashing does not need as much tempdb disk or procedure cache
  - Hashing will use a *lot* more tempdb data cache
  - May also result in fewer steps (up to 50% fewer)
- sp_configure "max resource granularity"
  - Among other uses, restricts how much tempdb buffer cache is used
- Always test and time it (and review regularly, else systems entropy!)

Update statistics (13 of 14)

Command(s)	Number of scans	Stats on a	Stats on b	Stats on c	Stats on d	Stats on e
update statistics T	1	Y	Y	N	N	N
update index statistics T	9 + 5 sorts	Y	Y	Y	Y	N
update all statistics T	5 + 3 sorts	Y	Y	Y	Y	Y
update all statistics T ... with hashing	1	Y	Y	Y	Y	Y
update statistics T (a, b, c, d)	???	Y	Y	Y	Y	N

- Wait, what now?
- What do you mean you don't know?!

Update statistics (14 of 14)

- It's right there in the manuals:

  If a comma-separated list, (col1), (col2)..., is used and hashing is enabled, then one scan can be used to gather statistics, if you do not exceed the resource granularity. If sorting is enabled, then one scan for each is used. If you use partial hashing, one scan can be used for low-domain columns, if you do not exceed the resource granularity, and one scan is used for the sort for each of the high-domain columns (that is, if you have three columns, you have three sorts).

  [...]

  Specifying more than one column in a column list (for example (col1, col2, ...)) generates or updates a histogram for the first column, and density statistics for all prefix subsets of the list of columns. Hash-based statistics cannot be used in this case.

Reorg (1 of 5)

- Could make a whole presentation just for this…
- reorg forwarded_rows / reclaim_space
  - Don't bother, stop running these
  - Use instead reorg compact
    - » Does the work of both in the time of either

Reorg (2 of 5)

- reorg rebuild
  - Stop running this today unless you have no choice
- Use instead reorg rebuild … with online
  - Requires a unique index
  - But you should have at least one of these anyway…

Reorg (3 of 5)

- Use reorg rebuild < table > < index >
  - Doesn't do data pages, but …
    - » Doesn't require select into
    - » Doesn't block dump tran
    - » Not run in a single transaction
    - » Doesn't need much free space in database

Reorg (4 of 5)

- Use reorg defrag
  - Requires DOL, not available for APL
  - Can run in timed batches
    - » … with time = HH:MM
    - » … with resume
      - picks up from where the last one stopped
  - … with skip_compact_extents=<percentage> (default = 80)

Reorg (5 of 5)

- As with update stats, run only where needed
- Look at DPCR and IPCR
- Interpret in context of row count
  - DPCR=0.95 is fine for 1000 rows but concerning for 1B+ rows
- Reorg defrag now can work only on parts of table needed

Create index (1 of 1)

- create index … with statistics
  - Get an update stats for free
- create index … with online
  - No longer holds blocking locks
  - Will still flood data caches and consume resources
- If creating multiple indexes, always create (APL) clustered first
  - Not an issue for DOL

ASE backups (1 of 1)

- Compressed backups are almost always a good idea
  - Even if you don't care about space
  - Fewer I/Os is well worth small increase to CPU
- Use the "newer" syntax
  - dump … with compression = NN
  - Internally more efficient than "compress::[…]"

ASE patching (1 of 1)

- Standard recipe for applying an ASE patch
  - apply/copy binaries and other files
  - start ASE (upgrades databases as they are brought online)
  - apply many scripts one at a time
- Easy to forget a script in the last step
- Check using sp_version
- Instead use the new *updatease* command line tool
  - Applies all of the scripts for you, in the correct order

Part Three
ASE Feature Insert List
aka "Make your life better"

ASE GUI DBA tools (1 of 2)

- Command line = scriptable = automatable = less toil = less error
- But there's no accounting for taste
- With Flash dead, the new ASE DBA GUI tool is AMC
  - Administration & Management Console
  - Uses dedicated login (sapsa) and database (saptools)
  - Intended to be used with built-in Job Scheduler

ASE GUI DBA tools (2 of 2)

- Automated Table Maintenance (update stats, reorg)
  - based on rules, thresholds, profiles
- Intended to be the front-end to the Workload Analyzer feature
- AMC is licensed as part of the ASE server, not the ASE client

ASE backups, again (1 of 2)

- Some features to reduce toil and technical debt
- Dump config stored and reused
- Dump history
  - Combine with sybrestore tool for correct sequence and no gaps
- sybdumptran tool
  - "ASE is dead, take special no-data dump tran anyway"

ASE backups, again (2 of 2)

- Some sp_configure parameters to make your life better
  - "enable concurrent dump tran"
  - "enable delta dump tran"
  - "optimize dump for fast load"
  - (TEST!) "enable buffered i/o for load"

ASE trace flags (1 of 2)

- If you promise to review as part of every patch…
- -T720 prevent "proc_hrd memory mismatch" spurious errors
- -T900 allow logins with oper_role to access "dbo use only" dbs
- -T1198 where possible don't use SH_INT locks for row_count()
- -T2512 skip syslogs during dbcc checkalloc
- -T2781 check column defaults are correct datatype
- -T3000 writes dump database start and end times

  add -T3605 to write to ASE error log

ASE trace flags (2 of 2)

- If you promise to review as part of every patch…
- -T5103 allow disk resize to shrink devices
  (requires 16.0 SP02 PL08+ or SP03 PL06+)
- -T7786 don't use statement cache for queries with large IN lists
- -T7790 don't use statement cache for statements > 16Kb
- -T16939 suppress showplan output for all built-in system procs
  (requires 16.0 SP02 PL05+)

Q&A, and thank you

Joe Woodhouse
joe.woodhouse@primadonnaconsulting.com

Too busy putting out fires to reduce your toil?
Answering the on-call phone too often?
Drowning in technical debt?

Joe is a freelance consultant available through Prima Donna Consulting and can be engaged nimbly and flexibly.

Would you like this or another of Joe's papers presented to your workplace? Recent topics:

- ASE Tips & Tricks
- ASE Tempdb performance & tuning
- ASE Memscale – use cases & benchmarks
- ASE Anti-Patterns

PRIMA DONNA CONSULTING