ISUG-TECH 2015 Conference

ASE Semantic Partitions – Two Case Studies

Joe Woodhouse

Agenda

- Welcome
- Speaker Introduction
- ASE Semantic Partitions
- Q&A

Joe Woodhouse

- joe.woodhouse@primadonnaconsulting.com
- Worked for Sybase Australia 1996 – 2003
- Freelancer since 2003
- Specialises in ASE, IQ, & RepServer
- Not a lawyer
  - Doesn't charge for emails or phone calls
- Hair changes colour every 3-4 weeks

Agenda

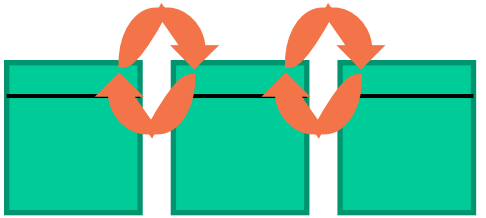
- Semantic Partitions – Overview and Implications
- The Sales Promise
- Two Case Studies and Findings
- Limitations
- Issues
- ASE 16.0+

Agenda

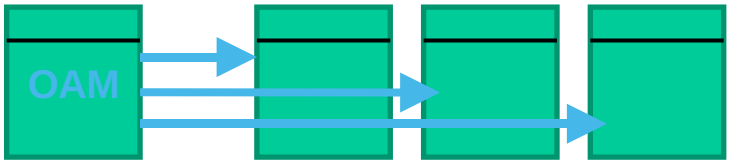
- **Semantic Partitions – Overview and Implications**
- The Sales Promise
- Two Case Studies and Findings
- Limitations
- Issues
- ASE 16.0+

Semantic Partitions Overview

- Review: ASE table data structures
- APL tables = list of pages
  - doubly-linked page chain



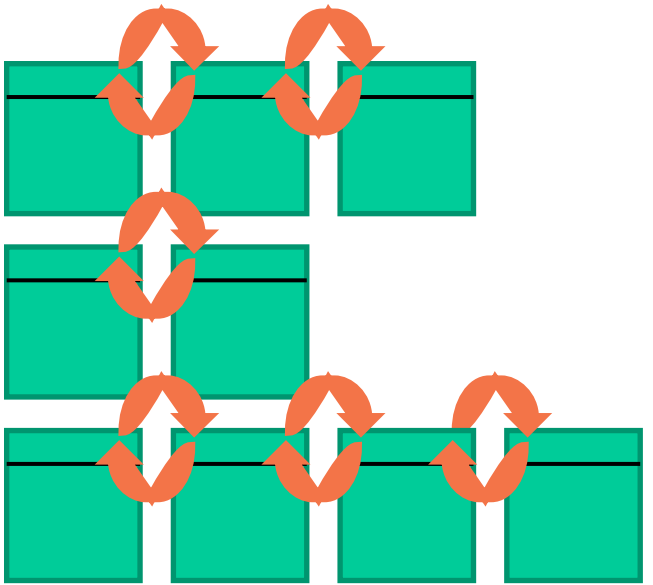
  - DOL tables = list of pages
  - no page linkage, linked from OAM page(s)



  - But either way, just one page chain

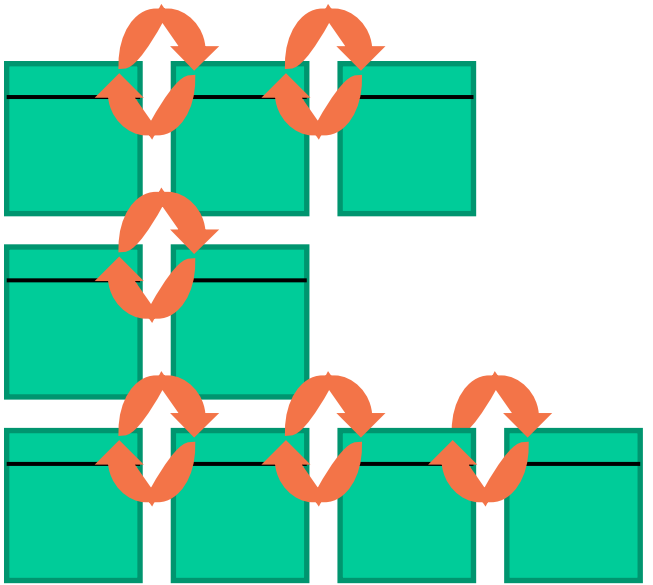
Semantic Partitions Overview

- Review: ASE round-robin partitioning
  - Available since SQL Server 11.0
  - Motivation: performance
  - Last page was hot spot on heap tables
  - So have more than one last page
  - Multiple page chains



Semantic Partitions Overview

- Review: ASE round-robin partitioning
- Partition balance was not maintained
- Manual maintenance
  - bcp into specific partitions
  - drop & recreate clustered index
- ASE 11.5 introduced parallel query
  - Issue with partition balance



Semantic Partitions Overview

- Question:

  Is round-robin partitioning the best answer today for last-page contention on heap tables?

Semantic Partitions Overview

- Question:

  Is round-robin partitioning the best answer today for last-page contention on heap tables?
- Answers:
  - Clustered index – no more heap table
  - But ascending key still forces last page insert
  - Data-rows locking scheme – no more page contention
  - Round-robin partitioning may still be a good answer

Semantic Partitions Overview

- ASE 15.0 introduced semantic partitioning
  - Semantic = “meaning”
  - Partition the data meaningfully rather than randomly
- Round Robin: no change from 11.0.x (not semantic)
- Hash: “distribute pseudo-randomly”
- List: “I’ll tell you which values go in which partition”
- Range: “values in this range go in this partition”

Semantic Partitions Overview

- All partition styles except round-robin are maintained
  - Updating a row may move it to another partition
  - Partitions are only as balanced as your design
    - Hash partitions will start and remain very balanced
  - All partition styles except round-robin are separately licensed
    - Not cheap, but look into Developer's Edition to play with it
  - All well documented in the (free) manuals
    - No surprises in how the feature works

Agenda

- Semantic Partitions – Overview and Implications
- **The Sales Promise**
- Two Case Studies and Findings
- Limitations
- Issues
- ASE 16.0+

The Sales Promise

- Query performance!
- (Better) parallel query!
- Partition elimination!
- Housekeeping performance!
- Update stats, dbcc, create index & reorg one partition at a time!
- Keep static data in archive partition – no more housekeeping!
- Together these give us … Scalability!

The Sales Promise

- The promises *are* real and *can* be achieved
- But not for everyone, and not all the time
- Seemingly no issue with *implementation*
- When it can be used, semantic partitions does what it says on the tin
- Not aware of any bugs that would keep it out of Production
- The issue is *design*
  - Limitations on when it can be used

Agenda

- Semantic Partitions – Overview and Implications
- The Sales Promise
- **Two Case Studies and Findings**
- Limitations
- Issues
- ASE 16.0+

Two Case Studies

- Case Study #1
- Client in the finance industry
- Motivation: query performance, “let’s buy Semantic Partitions!”
- Involved upgrade from ASE 12.5.4 to ASE 15.0.2
- “Hey, let’s change the hardware and O/S too”
- This is when they brought me in

Two Case Studies

- Case Study #1
- Client in the finance industry
- Motivation: query performance, “let’s buy Semantic Partitions!”
- Involved upgrade from ASE 12.5.4 to ASE 15.0.3
- “Hey, let’s change the hardware and O/S too”
- This is when they brought me in

Two Case Studies

- Database was downstream copy used for reporting purposes
- Semantic partitioning chosen for central fact tables
- Intention was to leverage parallel query and partition elimination
- Partitioning implementation was easy
- Cross-platform dump & load including upgrade was painful
  - Most of our issues encountered here

Two Case Studies

- Regression testing (new hardware, O/S, ASE but unpartitioned)
  - Results were not promising, worse than before
- Further testing with partitioning
  - Even worse! Why?
- Project was planned and budgeted based on Account Manager assurance that it was a trivial upgrade

Two Case Studies

- Regression testing (new hardware, O/S, ASE but unpartitioned)
  - Results were not promising, worse than before
- Further testing with partitioning
  - Even worse! Why?
- Project was planned and budgeted based on Ac
  Manager assurance that it was a trivial upgrad

Two Case Studies

- Remaining budget shifted to performance & tuning exercise
  - ASE 15.0.3 fixed a lot of issues
  - P&T fixed a lot more
  - O/S patches & tuning fixed more
- Initial small budget (“trivial upgrade”, remember) was exhausted
  - No appetite for Semantic Partitions once performance was improved vs. old system
  - “Good enough, we’ll call that a win”
  - Semantic Partitions license still sitting on a shelf

Two Case Studies

- Case study #2
  - A (different) client in the finance industry
  - Motivation: scalability, housekeeping performance
  - “Hey, here’s our design for semantic partitioning, make it work”

Two Case Studies

- Case study #2
  - A (different) client in the finance industry
  - Motivation: scalability, housekeeping performance
  - “Hey, here’s our design for semantic partitioning, make it work”

Two Case Studies

- Some lessons learned from the last attempt
- Validate the design before proceeding with implementation
- “Uhh… this isn’t going to work”
- Serious loss of face for Solutions Architect & Lead Designer
  - “Find another justification”
- “Hey, we could combine it with ASE 15.7 Compression for tiered storage”

Two Case Studies

- ASE 15.7 Compression POC very promising, but…
- No appetite for an ASE upgrade
- “We’ll park that and come back to it later”
- At least no money had been spent on the license

Two Case Studies

- So what went wrong? Is Semantic Partitioning broken?
- Not at all, it's a fine feature
- But the use cases are poorly understood and greatly over-sold
- It all came down to *indexing*

Agenda

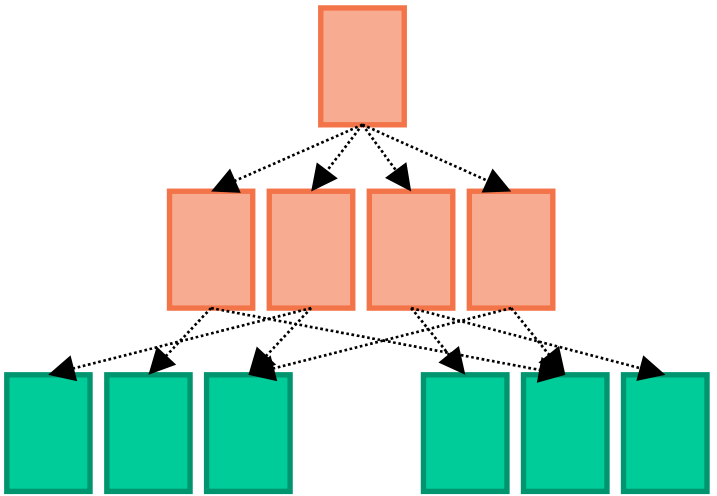
- Semantic Partitions – Overview and Implications
- The Sales Promise
- Two Case Studies and Findings
- **Limitations**
- Issues
- ASE 16.0+

Limitations

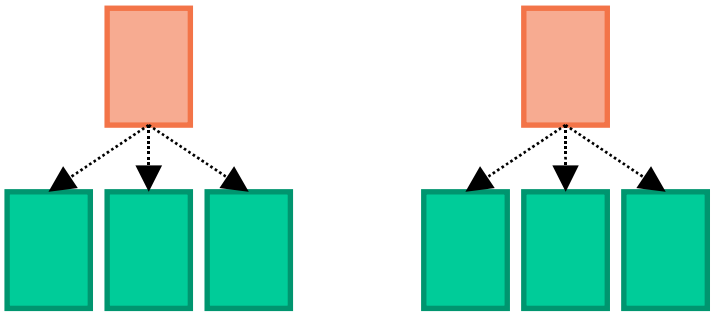
- Semantic Partitioning relies on a *partition key*
  - This defines which row goes in which partition
  - Not needed for round-robin partitioning (because not semantic)
- Partition key may be single-column or composite
  - Up to 31 columns for range and hash partitions
  - Single-column only for list partitions
- Partition key may be most data types
  - Not bits, BLOBs or derived columns

Limitations

- Global index applies to entire table, all partitions



- Local index is partitioned just like the data



Limitations

- Global indexes have limitations
  - Global + nonclustered allowed on all partition types
  - Global + clustered only on round-robin or unpartitioned tables
    - Uh oh, issue already if your schema uses clustered indexes
- Local indexes look great on paper
  - *Much* smaller, fewer levels, fewer I/Os
  - Partition elimination mean only ever read a subset of data
  - Housekeeping (update stats, reorg etc.) *much* faster

Limitations

- Ok, so we just convert everything to local indexes right?
  - Yes, if you want to realise all partitioning benefits, but…
  - Not so fast!
  - What about uniqueness?
- *Global* uniqueness in local indexes is not guaranteed
  - Same PK could occur in more than one partition

Limitations

- Local index uniqueness guaranteed only when
  - Partition key columns are a subset of the unique index columns
  - Partition key columns are in the same order as in the unique index
- Huh?

Limitations

- Table is partitioned by range, hash or list
  - Partition key (colA), local index key (colA) = can be unique
  - Partition key (colA), local index key (colB) = *cannot be unique*
  - Partition key (colA), local index key (colA, colB) = can be unique
  - Partition key (colA, colC)
    - Local index key (colA) = *cannot be unique*
    - Local index key (colC, colA) = *cannot be unique*
    - Local index key (colA, colB, colC) = can be unique

Limitations

- Even if you can use entirely local indexes…
- Does your partition key appears in your WHERE clauses?
- If so – great, you'll get good partition elimination and query efficiency
- If not – oops
  - How does ASE know which partition(s) your rows are in?
  - If no global indexes, the only way to find out is to read every local index
  - This may be a performance regression

Agenda

- Semantic Partitions – Overview and Implications
- The Sales Promise
- Two Case Studies and Findings
- Limitations
- **Issues**
- ASE 16.0+

Issues

- So what's the issue?
  - Your partitioning design probably doesn't satisfy these restrictions
    - You probably want range partitioning by date, or list partitioning by low cardinality key (State or region?)
    - These may not be in your unique or primary key
  - Your application probably requires a unique or PK index
  - Your partition key(s) may not be in your WHERE clauses

Issues

- Ok, add partition key(s) to your unique or PK index
  - If you can do this, everything is ok, but if not…
- Ok, give up on uniqueness
  - Unlikely to be possible! So…
- Ok, give up on local indexing
  - Means giving up (most) query performance
    - No partition elimination, but parallel query still possible
  - Means giving up housekeeping performance

Issues

- Even if you can deal with these design constraints
  - If partition keys don’t let you eliminate partitions, every data access must touch every partition
  - Query performance may regress
  - This was the issue with case study #1
- If you must have uniqueness, clustered indexes, but can’t change index keys
  - You might not be able to use semantic partitioning at all
  - This was the issue with case study #2

Agenda

- Semantic Partitions – Overview and Implications
- The Sales Promise
- Two Case Studies and Findings
- Limitations
- Issues
- **ASE 16.0+**

ASE 16.0+

- ASE 15.7 SP130 allows “partial indexes” on partitioned tables
  - Create local indexes in parallel on different (sets of) partitions
  - Create local indexes on only *some* partitions
  - Only for nonclustered local indexes

ASE 16.0+

- ASE 16.0 introduced partition locks
  - Lock one partition at a time for DML *and* DDL
  - Enable per table (via sp_chgattribute)
  - Configurable lock promotion
  - New sp_configure parameter “enable utility lvl 0 scan wait”
    - Add and drop partitions during isolation level 0 reads
  - ASE 16.0 index compression also available for local indexes

Conclusion

- Implementation is sound
- Not aware of any bugs
- Good design can realise all claimed benefits
- Good design simply may not be possible
- Unlikely to be a turn-key implementation
- May require significant database and/or application redesign
- All issues cited remain true even in ASE 15.7 SP100
  - Have not seen reported fixed in ASE 15.7 SP130 or ASE 16.0 SP01

Questions and Answers

Thank You for Attending

Please complete your session feedback form