

SQLIntersection PRECON02

What's New in SQL Server 2014

Bob Beauchemin
bobb@SQLskills.com



About Bob Beauchemin

- **Independent Consultant/Trainer/Writer/Speaker**
- **Developer Resources Partner, SQLskills**
 - Website: <http://www.SQLskills.com>
 - Blog: <http://www.SQLskills.com/blogs/bobb>
 - Email: bobb@SQLskills.com
 - Twitter: @bobbbeauch
- **SQL Server MVP**
- **Author of books and numerous resources related to SQL Server and data access**
 - Microsoft SQL Server 2012 Internals (special indexes chapter)
 - SQL Server MVP Deep Dives – Volume 1
 - A Developer's Guide to SQL Server 2005
 - A First Look at SQL Server 2005 for Developers
 - Essential ADO.NET



SQLintersection

Orlando, Florida – April 2014

www.SQLintersection.com

Outline

- In-Memory OLTP
- Columnstore Improvements
- New Cardinality Estimator
- Additional Performance Features
- Azure Integration
- Others



3

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

1- In-Memory OLTP

- Pillars of In-Memory OLTP
- Memory-Optimized Tables Details
- Compiled Stored Procedures Details
- Multi-version concurrency control
- Programming differences
- Operational differences
- Migrating



4

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



2-Columnstore Enhancements

- **Columnstores**
 - Column-based storage vs. Row-based storage
 - Clustered columnstore
 - Archival compression
- **Query Improvements**
 - Batch mode
 - Partition and segment elimination
- **Updatable Columnstore**
 - Deltastores
 - Insert batch improvement
- **Use cases**
 - For clustered and nonclustered columnstore



5

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

3-Cardinality Estimation

- **Cardinality Defined**
- **Cardinality Estimator – SQL Server 7 – 2012**
- **Cardinality Estimator – SQL Server 2014**
- **Tracing and Troubleshooting the new Cardinality Estimator**



6

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

4-Additional New Features

- Performance Features
- Reliability and Security Features
- Hybrid Cloud Features



7

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQLintersection PRECON02

What's New in SQL Server 2014 In-Memory OLTP

Bob Beauchemin
bobb@SQLskills.com



Outline

- **Pillars of In-Memory OLTP**
- **Memory-Optimized Tables Details**
- **Compiled Stored Procedures Details**
- **Multi-version concurrency control**
- **Programming differences**
- **Operational differences**
- **Migrating**



9

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

OLTP and OLAP

- **OLTP and OLAP are general categories for data applications**
 - OLTP applications
 - Read one or very few rows at a time
 - Read all or many of the columns in a row
 - OLAP applications
 - Reads many rows – scans or range scans
 - Reads very few columns from a row
- **SQL Server 2014 introduces technologies to speed up OLTP systems**
 - Called In-Memory OLTP
 - Formerly called Hekaton
 - Also known as XTP (Extreme Transaction Processing)



10

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Pillars of In-Memory OLTP

- **Memory-optimized tables**
 - Separate storage engine
- **Compiled stored procedures**
 - Separate procedure and query processor
- **Lock and latch-free concurrency transaction processing**
 - Separate transaction management
- **Built in to SQL Server**



11

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Components

- **XTP storage engine**
 - Storage management
 - Hash and range indexes on tables
 - Transactional operations
 - Checkpoint
 - Recovery
 - High Availability
- **XTP compiler**
 - Compiles procedures into native code
 - Produces native access routines for in-memory objects
- **XTP runtime system**
 - Integration with SQL Server resources
 - Library of functions for compiled procedures



12

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Getting Started

- **You must have 64 bit SQL Server**
- **Database needs to have filegroup for memory-optimized data**
 - Only one filegroup for memory-optimized allowed
 - Filegroup can have multiple containers
 - Checkpoint data is stored in filestreams
- **Carefully consider collation**
 - Of your database
 - Of character-based columns
 - Learn the T-SQL COLLATE clause
- **Can consider**
 - MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT database option
 - Delayed durability



13

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Storage

- **Persistent memory-optimized objects stored as streams**
 - Also known as Checkpoint File Pairs (CFP)
 - Each CFP is one data file and one delta file
 - Limit: 4096 pairs in CTP2, 8192 in RTM
- **Data Files**
 - 128 MB chunks (RTM: 16 MB on machines with <= 16 GB physical memory)
 - 8 files initially allocated
 - Checkpoint-based sets of rows for all tables
- **Delta Files**
 - Stores IDs of deleted rows
- **Files are append-only**



14

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Memory-Optimized Tables

- **Tables and table variables**
 - Table variables must be strongly typed
- **Durable and non-durable tables**
 - Table variables always non-durable
- **Durable tables persisted into filestream-like filegroup**
 - Must define a special filegroup
 - Multiple containers supported
- **Durable tables are (re)-loaded at startup time**
 - Must have enough memory
- **Non-durable are not persisted or logged**
 - For ETL staging or temp table replacement



15

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Memory Considerations

- **Memory-optimized tables' memory is non-pageable**
 - Different memory clerk than buffer pool
 - Covered by max_server_memory
- **~80% hard limit for memory-optimized tables**
 - Changes fail after limit hit
- **Can also limit through resource governor**
 - Limit is on total amount per-pool allocated to XTP
 - Resource governor per-workload group memory limits do not apply to XTP
- **Memory is not allocated in pages**
 - Linked lists over hash buckets
 - Tables still have 8060 byte limit



16

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Table Layout and Versions

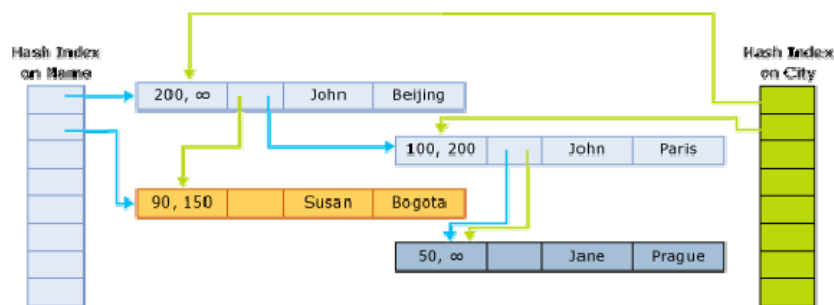
- **Uses a multi-version engine**
- **Each row has a BeginTimestamp and EndTimestamp**
- **Inserts**
 - Insert new rows with BeginTimestamp of tx timestamp, EndTime of infinity
- **Updates**
 - Insert new row - BeginTimestamp of tx timestamp, EndTimestamp of infinity
 - Set EndTimestamp of old row to tx timestamp
- **Deletes**
 - Sets EndTimestamp of old row to tx timestamp
- **Row access is based on statement/transaction begin time**
- **Old rows are garbage collected**
 - When no longer needed



17

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Memory-Optimized Tables



18

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Memory-Optimized Table Code

- **When a memory-optimized table is created a DLL is created**
 - One DLL per table with access methods
- **DLL is not stored in SQL Server**
 - In a subdirectory of ...\\Data
- **DLLs are recreated and reloaded at startup time**
 - For memory-optimized tables
- **Can be observed**
 - Creation and compilation with XEvent events
 - With sys.dm_os_loaded_modules



19

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Indexing

- **Two types of indexes supported**
 - Hash index – useful for point queries
 - Range index – useful for range queries
- **Up to 8 indexes per table**
- **Tables must have at least one index**
 - Durable tables also need primary key
 - Index columns must be non-null, need not be unique
- **Indexes must be declared at CREATE TABLE-time**
 - No CREATE/ALTER INDEX or ALTER TABLE
 - Fragmentation and fill factor do not apply
- **Indexes are not stored on disk**
 - Created at load time
- **All indexes are covering**
 - Because they are pointer-based



20

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Hash Indexes

- **Useful for point queries only**
 - Equality or IN predicates
- **Number of hash buckets must be declared with table**
 - 8-byte pointers
 - Should be 2x expected number of rows
 - Internally rounded up to the next power of two
 - Hard limit of 1,073,741,824 buckets
- **Hash collisions will chain rows, decrease performance**
 - Increase bucket_count
- **Less useful for composite keys**
 - Whole key must match



21

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Range Indexes

- **Range indexes are declared as NONCLUSTERED indexes**
 - Uses new BW-tree – lock and latch free
- **Range indexes are unidirectional**
 - Do not support backward scans
- **Pages are stored differently**
 - Not fixed size, but still maximum 8k
- **Range indexes handle updates differently than B-tree**
 - Index pages are never updated
 - They are replaced with a new page and mapping table is updated
 - Atomic replace uses compare-and-swap machine instruction
 - Each insert, update, delete to key value on that page
 - Produces a page w/delta record indicating the change that was made.



22

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Compiled Procedures

- **Compiled at CREATE PROCEDURE-time**
 - T-SQL is translated to C code
 - Native DLL produced
- **Re-created on first use after system shutdown**
- **Query plan is created at CREATE PROCEDURE-time**
 - Statistics should be update before procedure creation
 - No auto-statistics update
 - No ALTER PROCEDURE, CREATE... WITH RECOMPILE, etc
- **Compiled procedures can only access memory-optimized tables**
 - Not disk-based (“traditional”) tables
- **Compiled code generated for memory-optimized tables**
 - Access routines



23

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

CREATE PROCEDURE declarations

- **Compiled procedures must be declared with**
 - NATIVE_COMPILATION
 - SCHEMABINDING
 - EXECUTE AS OWNER/SELF/user (EXECUTE AS CALLER not allowed)
 - BEGIN ATOMIC
- **The BEGIN ATOMIC statement declares**
 - Transaction isolation level
 - Language
 - DATEFORMAT, DATEFIRST – optional
 - DELAYED_DURABILITY = ON | OFF
- **Can specify parameters and variables as NOT NULL**
 - Compiled procedures only



24

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Atomic Blocks

- **Each compiled procedure is an atomic block**
 - Cannot begin or end transactions in compiled procedures
- **It's transactional standalone or composed in user transactions**
- **Composition with user transactions**
 - Accomplished with savepoints, not autonomous transactions
- **Errors during procedure execution will rollback procedure code**
- **TRY-CATCH to limit rollback**
 - But no deadlocks occur
- **Transactions, procedures should be kept short**

Compiled Procedure Use Cases

- **More procedural code (and more rows) that are processed, the bigger the benefit from compiled procedures**
 - Aggregation
 - Nested-loops joins
 - Multi-statement select/insert/update/delete
 - Complex expressions
 - Procedural logic, like conditionals and loops

Troubleshooting

- **Code is never auto-recompiled**
 - Update statistics before CREATE
 - Must drop and recreate to update plan
- **Code consists of**
 - Table IO from per-table DDLs
 - Built-in functions code
- **Optional DMV information**
 - Turn on query_stats and procedure_stats info on demand
 - Query_stats can be per-procedure, procedure_stats is global
 - Only timings, execution counts, last compiled/executed
- **No T-SQL debugging**



27

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Transactions and Isolation

- **In-Memory OLTP uses multi-version engine**
 - MVCC = Multi-version concurrency control
 - Multiple versions of rows in memory
 - Only one version is valid at a given time
 - Which one is read depends transaction's begin time
- **For access from interpreted SQL**
 - MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT on CREATE DATABASE
 - READ COMMITTED supported for single statements
 - Otherwise statements need isolation level hints
- **Optimistic concurrency used**
 - No locks and no latches
 - Assumes other transactions will commit (doesn't wait)
- **Error-handling and retry logic required**



28

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Isolation Levels

- **SNAPSHOT**
 - Read-consistency as of beginning of transaction
- **REPEATABLE READ**
 - Row that are read cannot change from beginning to end of tx
- **SERIALIZABLE**
 - All scans are repeatable too
 - No phantoms
- **Write-write conflicts cause rollback**
 - Immediately
- **Violations of isolation level cause rollback at commit time**
 - In prepare phase

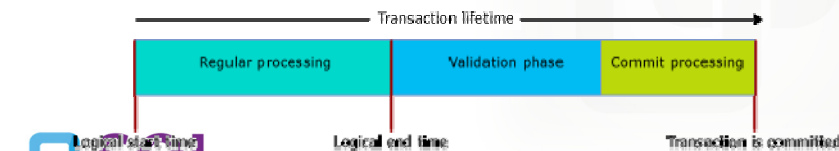


29

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Transaction Phases

- **Transactions consist of three phases**
- **Transaction created – begin timestamp**
 - Normal Processing
 - Preparation
 - Gets end timestamp
 - Must resolve transaction dependencies
 - Checks for concurrency violations
 - Postprocessing
 - Writes end timestamp
- **Transaction terminated**



30

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Isolation Level Implementation

- **Each transaction that updates data keeps WRITE_SET**
 - Used to update timestamps quickly when transaction commits
- **Repeatable read transactions use READ_SET**
 - Used to check that rows have not changed between transaction logical begin time and logical end time
 - Rows reread at validation time
- **Serializable transactions use READ_SET and SCAN_SET**
 - Validates reads like repeatable read
 - Validates that no phantom inserts occurred between transaction logical begin time and logical end time
 - Ranges re-scanned at validation time



31

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Speculative Reads

- **If another transaction is in progress**
 - Readers assume other transaction will commit
 - Can do speculative reads
 - Can speculate and ignore versions
- **This causes transaction dependencies**
- **Must be resolved before proceeding**
 - Can cause waiting here



32

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Garbage Collection

- **Two kinds of garbage collection**
 - Garbage collection of rows in memory
 - Garbage collection of checkpoint data files (merge)
- **Garbage collection of rows in memory is cooperative**
 - User threads can collect old versions during scans
 - System GC process runs once per minute (or more)
 - Assigns GC queue items to user threads
- **Merge operation merges data files**
 - If two consecutive data files < 50% full
 - Merge can also be forced by system stored procedure



33

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

But there are limits...

- **Limits on memory-optimized tables**
 - Only some data types
 - BIN2 collations on columns with indexes
 - 8060 byte row limit
 - No constraints (except NULL), triggers
- **Limits on T-SQL statements**
 - TRUNCATE TABLE unsupported
- **Limits on T-SQL constructs usable in compiled procedures**
 - Compiled procedures can only access memory-optimized tables
 - Compiled procedures can't call other procedures
 - Some constructs (e.g. OUTER JOIN) unavailable
 - Many functions unavailable
 - ORDER BY and comparison – BIN2 collations only
- **Limits on transaction isolation levels**
 - And limits on cross-container transactions



34

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Programming Differences

- **Uniqueidentifier is a useful hash key**
 - No worries about index fragmentation
 - Sequence object can be a bottleneck
- **You must account for errors**
 - Transactions more likely to roll back
- **No parallelism for query referencing memory-optimized tables**
- **Deleting a range of rows with concurrent insert/update will likely fail**
- **All parameters for native procs assumed to have unknown values**
 - OPTIMIZE FOR hint is useful more often



35

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Programming Differences (2)

- **Design changes**
 - No constraints or triggers – integrity redesign
 - No large data types – table redesign
 - Hash index only used for point queries – index redesign
- **Smaller function library and query surface area**
 - You may need to write your own equivalents
 - Some pre-written examples



36

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Management Differences

- **No index rebuilds needed**
 - Indexes rebuilt at startup
- **No automatic statistics updates and query plan rebuilds**
 - Must manually rebuild stats and compiled sprocs
 - Must specify NORECOMPUTE and FULLSCAN
- **No schema changes without dropping existing**
 - No ALTER TABLE
 - No ALTER PROCEDURE
 - Compiled sprocs must be declared WITH SCHEMABINDING
- **Availability groups supported**
 - Not replication or database mirroring
 - Non-durable tables don't get replicated in AGs



37

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Management Differences (2)

- **Must have sufficient memory**
 - Must have strategy to deal with out-of-memory
 - XTP competes with the buffer pool
- **Table population is parallelized per-filegroup**
 - Multiple memory-optimized filegroups help
- **Log speed is the most limiting resource**
 - Some use cases for delayed durability
 - Logging is more efficient in general
- **Limit 256 GB – limit applies to durable tables**
- **Cannot drop memory-optimized filegroup**
- **Statement-level audit not supported in compiled sprocs**



38

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Instrumentation

- **Extended Events Categories**
 - Checkpoint
 - Deploy
 - GC
 - Transaction
 - XTP
- **DMVs**
 - Instance-wide – sys.dm_xtp...
 - Database-specific – sys.dm_db_xtp..
- **Perfmon Counters**
 - XTP series
- **Metadata**
 - sys.heap_indexes
 - New fields in sys.tables, sys.table_types, sys.procedures, others



39

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Migrating to In-Memory OLTP

- **Run AMR (Analysis, migrate, and support) Tool**
 - Set up Management Data Warehouse
 - Enable Data Collection Sets
 - Table Usage Analysis
 - Stored Procedure Usage Analysis
 - Run workload
 - Generated AMR Reports (MDW database/Reports)
 - Doesn't add much overhead to production server
 - Requires 2014 SSMS
- **Memory Optimization Advisor and Native Compilation Advisor**
- **Migrate Tables to Memory-Optimized Tables**
 - Workarounds for some common limitation in BOL
- **Migrate Procedures to Compiled Stored Procedures**
 - When possible



40

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Migration Best Practices

- **Get a baseline**
- **Test a realistic workload**
 - Not just batches of insert statements
 - Right transaction mix to test locking vs lock-free
 - Use distributed replay with test suites
- **It doesn't fix everything**
 - More than 60 cores may affect performance



41

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Summary

- **In-Memory OLTP feature consists of**
 - Memory-optimized tables
 - Multi-version engine
 - Hash and range indexes
 - New, pointer-based layout (no pages)
 - Persistent schema_and_data or just schema persistent
 - DLL-per-table for data access
 - Compiled stored procedures
 - T-SQL compiled into C code, compiled into DLLs
 - Atomic blocks
 - Error-based rollback
 - Declarative isolation level and other settings
 - Lock and latch-free access
 - Multi-version engine
 - Multiple isolation levels supported
- **Part of SQL Server**
 - Not a separate add-on



42

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Resources

- **Books Online – In Memory OLTP section**
- **Whitepapers**
 - SQL Server In-Memory OLTP Internals Overview for CTP2
 - Heketon: SQL Server's In-Memory OLTP Engine (Sigmod 2013)
 - High-Performance Concurrency Control Mechanisms For Main-Memory Databases
 - The BW-Tree: A B-tree for New Hardware Platforms
- **Blogs**
 - SQL Server Team Blog
 - My blog
 - Tony Rogerson's Rambling on Data
 - Blakhani – Help: SQL Server - A-Z of In-Memory OLTP



43

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

SQLIntersection PRECON02

What's New in SQL Server 2014 Columnstore Enhancements

Bob Beauchemin
bobb@SQLskills.com



Introduction

- **Columnstores**
 - Column-based storage vs. Row-based storage
 - Clustered columnstore
 - Archival compression
- **Query Improvements**
 - Batch mode
 - Partition and segment elimination
- **Updatable Columnstore**
 - Deltastores
 - Insert batch improvement
- **Use cases**
 - For clustered and nonclustered columnstore



45

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

OLAP and OLTP

- **OLAP and OLTP are general categories for data applications**
 - OLTP applications
 - Read one or very few rows at a time
 - Read all or many of the columns in a row
 - OLAP applications
 - Reads many rows – scans or range scans
 - Reads very few columns from a row
- **Columnstores are a good choice for OLAP applications**
- **Columnstores are significantly slower for OLTP applications**



46

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Columnstore and Data Warehouse

- **Data warehouse is the primary use case**
- **Data warehouse star or snowflake schema consists of**
 - Fact Tables – large number of rows, wide rows
 - Dimension Tables – Small tables to group by
- **Fact Tables are notoriously difficult to index**
 - Aggregates are pre-calculated offline for speed
 - Columnstores help here
 - May also help with large dimension tables



47

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

xVelocity

- **Based on xVelocity technology**
 - Introduced in PowerPivot (was Vertipac)
 - In SQL Server 2012, [implemented in database engine](#) and SSAS
- **Profound speedup for data warehouse queries**
 - 10-100 times faster
 - Not all queries benefit



48

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Column-Based Storage

- **SQL Server data is customarily stored in rows**
 - Rows are stored in pages
 - Pages hold 1-n rows
- **Columnstore data is stored on a per-column basis**
 - Column segments contain values from the same column
 - Segment is the unit of transfer between disk and memory
 - Rowgroups refer to sets of column segments from columns in same table
 - Technically, column segments are stored as blobs
 - And data is still stored in pages
- **Columnstore is an Enterprise feature**

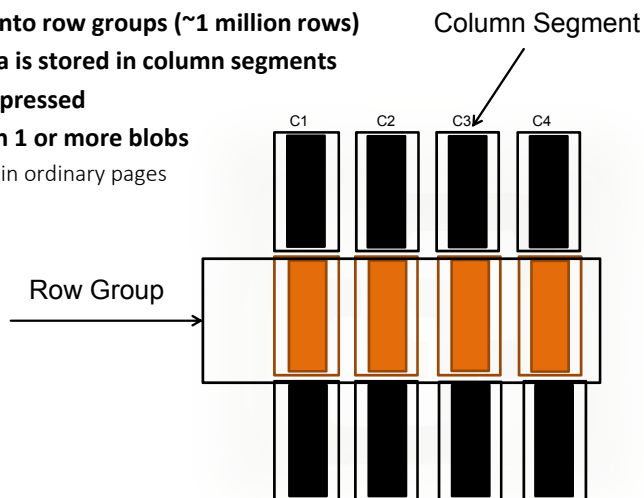


49

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Row Groups and Column Segments

- Rows are divided into row groups (~1 million rows)
- Each column's data is stored in column segments
- Segments are compressed
- Segments stored in 1 or more blobs
 - Blobs are stored in ordinary pages



50

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Columnstore Indexes

- **Columnstores are known as “columnstore indexes” (CCI)**
- **If columnstore is the only storage,**
 - This is “clustered columnstore index”
 - Not related conceptually to clustered B-tree index
 - Clustered columnstore index new in SQL Server 2014
- **If both row store and a columnstore exist (NCCI)**
 - This is “nonclustered columnstore index”
 - Row store can contain columns not in columnstore
 - Introduced in SQL Server 2012
- **Either type of index can be partitioned**



51

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Building Columnstores

- **To build clustered columnstore index**
 - Create heap or table
 - Or use existing table
 - Drop nonclustered indexes (if any) and constraints
 - All column data types must be supported by columnstore index
 - CREATE CLUSTERED COLUMNSTORE INDEX
 - ...WITH DROP_EXISTING = TRUE
 - Uses all columns in table, all column types must be supported
- **To build nonclustered columnstore index**
 - CREATE COLUMNSTORE INDEX
 - With all columns or subset of columns
 - One columnstore index/table
 - Must be partition-aligned if used with partitioning
 - Makes table read-only



52

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Data Type Support

- **Unsupported Types**
 - Decimal with precision greater than 18
 - Binary
 - BLOB
 - (n)varchar(max)
 - Uniqueidentifier
 - Datetimeoffset with precision greater than 2
 - CLR (includes spatial)
- **SQL Server 2014 includes**
 - All decimal
 - All datetimeoffset
 - Uniqueidentifier
 - Binary, varbinary



53

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Compression

- **Various compression techniques used**
 - Dictionary Encoding
 - Run-length Encoding
 - Huffman Encoding
 - Lempel-Ziv-Welch
 - Value Encoding
- **Archival compression in SQL Server 2014**
 - Add another layer of compression using XPress8 (a variant of LZ77)
- **Dictionary used for:**
 - All string columns
 - Non-string columns with few distinct values
- **Two types of dictionary**
 - Primary: One per column
 - Created only during index build
 - Secondary (overflow): 0-nper column
 - Each segment has 0 or 1 secondary dictionary
 - Secondary dictionary may be used by more segments



54

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Metadata Tables

- **sys.column_store_segments**
 - Shows one row per segment / per index partition
- **sys.column_store_dictionaries**
 - If dictionaries used for columnstore compression
- **sys.column_store_row_groups (SQL Server 2014 only)**
 - Can distinguish between deltastore and segment
 - Shows status and status description



55

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Performance Improvements

- **Storage format**
 - Columnstores reduce IO
 - Smart compression and encoding reduce memory and disk utilization
 - Improved buffer pool usage
- **Query Improvements**
 - Uses less memory during queries
 - Batch mode query iterators



56

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server 2014 Changes

- **Columnstore can be the only storage for a table**
 - Known as clustered columnstore index
 - If so, that table is updateable
 - Updates use deltastore (more later)
 - Schema can be changed
- **For clustered and nonclustered columnstore**
 - Additional data compression type
 - Columnstore archive compression
 - More data types are supported
 - Improved global dictionaries for better compression
 - Short string support without using dictionaries



57

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Columnstore and Queries

- **Optimizer is still cost-based**
 - Optimizer can choose index
 - Optimizer can choose batch mode
 - Batch mode not always used
- **Can observe in query plan**
 - Columnstore index use
 - Batch execution mode
- **Columnstore queries can benefit from**
 - Segment elimination
 - Partition elimination (if partitioning used)
- **Some queries slower with columnstore**



58

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Batch Mode Execution

- **Query processor usually uses row mode**
 - Query operators process one row at a time
- **SQL Server 2012 added batch mode**
 - Also known as "vectorized execution"
 - Operators handle ~1000 rows at a time
 - Saves CPU resources - less instructions executed
- **Batch mode can be observed in**
 - Query plan (on a per-operator basis)
 - Query execution speed



59

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Statement Improvements - SQL 2014

- **More iterators support batch mode**
- **Mixed-mode plans (batch and row) supported**
- **Seek operation support**
- **Bulk inserts optimized with clustered columnstore indexes**



60

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Batch Mode Restrictions – SQL Server 2012

- **Restrictions (in SQL Server 2012)**
 - Batch Processing iterators not used for
 - Outer joins
 - Union
 - Estimated/Actual execution mode = Row
 - NOT batch
 - May be able to rewrite query to use Batch Processing



61

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Query Hints (SQL Server 2012)

- **Nonclustered Columnstore-specific query hints**
 - USE columnstore index (traditional index hint)
 - USE non-columnstore index (traditional index hint)
 - OPTION(ignore_nonclustered_columnstore_index)



62

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server 2014 Batch Mode Improvements

- **Batch Mode Hash Join Support**
 - Improved batch-mode hash join handles inner, outer, semi and anti-semi joins
 - Batch mode hash joins handle spilling better
- **Union all supported**
 - But not union
- **Partial aggregation**
 - Global aggregation use row mode
 - Distinct aggregates use row mode



63

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Updating with Nonclustered Columnstore

- **Table cannot be updated directly**
- **During nightly batch processing**
 - Disable (or drop) columnstore index
 - Nightly load (update table)
 - Rebuild columnstore index
- **Table can also be "updated" via partition switching**
 - Load new data into staging table
 - Build columnstore index
 - Split empty partition
 - Switch the staging table partition into table
- **Variations possible for more frequent updates**
 - Trickle loading



64

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Updating Clustered Columnstore

- **Updates use 1-n deltastores**
 - This is a row store, not a columnstore
 - Insert – to deltastore
 - Delete – logical operation, data removed on next rebuild
 - Update – Insert and Delete pair
 - Bulk Insert – Goes to the columnstore if large enough (> 100k)
 - Otherwise, goes to delta store
 - Select – uses deltastore and columnstore (union internally)



65

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Deltastore Behavior

- **Disadvantages of Deltastore**
 - The entire row has to be read
 - Segment elimination does not occur for deltastores
 - Parallel scans will distribute the deltastores among the threads so that multiple deltastores are scanned in parallel
 - No inter-deltastore parallelism
 - Concurrent deletes or updates **are blocked** while the Tuple Mover compresses a deltastore.
 - Concurrent Inserts are not blocked by the Tuple Mover.
- **Delete bitmap**
 - Only compressed segments use a deleted bitmap



66

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Updating and Columnstore Segments

- **Conversion of rows (delta store) to columnstore uses tuple-mover**
 - Automatic when segment is full
 - Index reorganize starts it manually
 - Manual invoke results in undersized segments
- **Bulk Loading is better way to update**
 - Batches of less than ~100000 rows loaded as segments
 - Batches of less than ~100000 rows loaded into deltastore
- **Rebuilding index more useful**
 - Rebuilding index can make optimum size segments
 - Rebuilding index rebuilds the global dictionary
 - Reorganize index doesn't



67

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Better Memory Management – SQL Server 2014

- **When building columnstore**
 - Adjusts to low memory condition
- **Runtime**
 - Batch mode spilling not longer reverts to row mode
 - Presence of row operators doesn't prohibit batch mode
- **Honors resource governor**



68

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Use Cases and Best Practices

- **Usually good for fact tables in data warehouse**
 - Maybe large dimension tables
- **Fits with data warehouse pattern of**
 - Read-only tables used for analysis
 - Nightly job to append data
- **Queries aggregate lots of base rows**
- **Columnstore is relational table-based**
 - Can use PowerPivot
 - Can use SSAS instead (in SQL Server 2012)
- **Choose your xVelocity vehicle**



69

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

When to use Nonclustered (SQL Server 2014)

- **You must use nonclustered columnstore**
 - If you need key constraints or triggers
 - If you have unsupported data types
 - Consider refactoring
 - Consider refactoring for string types in any case
- **You should use clustered columnstore**
 - One copy of the data (you won't need other indexes)



70

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Review

- **Columnstore improvements**
 - Clustered columnstore
 - For nonclustered columnstore too
- **Archival compression**
- **Batch mode improvements**
 - For clustered and nonclustered queries
- **Deltastores and updating clustered columnstore**
 - Optimizing batch inserts
- **Memory management improvements**



71

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

References

- **SQL Server 2014 Books Online**
- **Columnstore Index Articles on Technet Wiki**
 - SQL Server Columnstore Index FAQ
 - <http://bit.ly/HMfRKw>
 - SQL Server Columnstore Performance Tuning
 - <http://bit.ly/wtOCLj>
 - Others...
- **Blogs**
 - Remus Rusanu
 - <http://rusanu.com/blog/>
 - Niko Neugebauer (series on clustered columnstore)
 - <http://bit.ly/1d8ykbB>



72

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQLintersection PRECON02

What's New in SQL Server 2014 Cardinality Estimation

Bob Beauchemin
bobb@SQLskills.com



Outline

- Cardinality Defined
- Cardinality Estimator – SQL Server 7 – 2012
- Cardinality Estimator – SQL Server 2014
- Tracing and Troubleshooting the new Cardinality Estimator



74

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Definition

- **Cardinality in databases**
 - The relationship between two tables (e.g. one-to-many)
 - The uniqueness of data values contained in a column
- **In query processor**
 - How many rows will satisfy this iterator
 - Query plan optimization depends on cardinality estimator
- **Query processor uses**
 - Statistics-based (with all the limitations of statistics)
 - Constraints (e.g. unique constraint)
 - Heuristics (when parameter values are unknown)
- Estimates are based on a set of basic assumptions



75

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Basic Assumptions

- **Uniformity**
 - Distinct values are evenly spaced and have same frequency
- **Containment**
 - For column = constant, assume that constant exists for column
 - For equijoin, values of foreign keys exist
- **Independence**
 - No dependence between columns in same table
 - $\text{Sel}(P \wedge Q) = \text{Sel}(P) * \text{Sel}(Q)$



76

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Estimation Example

- **For a simple WHERE clause with single column statistics**
 - RangeHighKey - best
 - AVG_RANGE_ROWS - interpolate between range
 - Density of entire column ($1/\text{\#distinct values}$)
 - Hardcoded guess (when parameterized and value is unknown)



77

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Old Cardinality Estimator

- **Introduced in SQL Server 7**
- **Extensions added over time**
 - Through trace flags
 - Through version-compatibility features
- **Over time it's hard to predict, understand, or control**
- **Customers hate plan regressions most of all**



78

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Cardinality Estimation Trace Flags

- **TF 2301 – Modeling Extensions**
 - Integer Modeling
 - Comprehensive Histogram Usage
 - Base Containment Assumption
 - Comprehensive Density Remapping
 - Comprehensive Density Matching
 - <http://blogs.msdn.com/b/ianjo/archive/2006/04/24/582219.aspx>
- **TF 2389/2390 - Handles ascending key problem**
 - Optimizer will brand a key ascending after a few statistics updates (2389)
 - Statistics auto-updated at compile time
 - 2390 assumes every key column without branding
 - <http://blogs.msdn.com/b/ianjo/archive/2006/04/24/582227.aspx>



79

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The Catch-All Trace Flag

- **TF 4199**
 - Incorporates all previous fixes made for query processor by trace flags
 - Not just cardinality estimates
 - Added in
 - SQL Server 2005 SP3 CU6
 - SQL Server 2008 and 2008 SP1 CU7
 - SQL Server 2008 R2 and SQL Server 2012
 - <http://support.microsoft.com/kb/974006/en-us> for list



80

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Tracing Inaccurate Cardinality Estimation

- **An XEvent added in SQL Server 2012**
 - `inaccurate_cardinality_estimate`
- **Fires on some iterators**
 - That produce different number of rows than they consume
 - Can fire multiple times per iterator and per query
- **Calculates an inaccuracy threshold at execution time**
 - Base on number of rows estimated
- **Fires (and recalculates threshold) when the threshold is reached**
 - Actual number of rows reflects the number returned so far by that iterator



81

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server 2014 Cardinality Estimator

- **Database compatibility level determines estimator**
 - Compatibility level 120 – new estimator
 - Older compatibility levels – old estimator
- **Trace flags can control it**
 - TF 2312 – use new estimator regardless of compatibility level
 - TF 9481 – disable new estimator
- **For individual regressions, use query hint**
 - `OPTION QUERYTRACEON(2312)`
- **Query plans show which cardinality estimator used**
 - `CardinalityEstimationModelVersion = 70/120` – statement iterator properties



82

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

New Estimator Goals

- **Consistent, predictable query performance**
 - Different operator trees that represent the same relation have the same cardinality estimates.
- **New model for better performance**
 - Significant changes to the model result in more accurate cardinality estimates and better plan choices
- **Incorporate special case code**
- **Easier to support**
- **Minimal regression**



83

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

How it's different

- **SQL Server 7 cardinality estimator**
 - Integrated in query optimization
 - Special cases in code
 - Two wrong estimates sometimes made a right one (but not usually)
- **SQL Server 2014 estimator works in two stages**
 - Choose appropriate estimation calculator (~50 choices)
 - Execute the estimation process with the appropriate statistics
- **SQL Server 2014 estimator includes refinements of basic assumptions**



84

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Refinements of Basic Assumptions

- **Independence vs. Correlation**
 - Old: Columns independent unless multicolumn stat density says otherwise
 - New: Use exponential backoff for multicolumn, single table predicate
- **Containment – simple join**
 - Old: Use histogram from each table
 - New: Assume keys in table A exist in table B
- **Containment – ascending key (New)**
 - Always assume query predicate value exists
 - Estimate cardinality using average frequency
- **Containment – multiple predicates in joined tables**
 - Old: Simple containment
 - All rows matching predicate in table A match predicate in table B
 - New: Base containment
 - All rows matching predicate in table A exist in table B (but don't always match)



85

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Tracing Cardinality Estimation

- **A single XEvent produces detailed output**
 - query_optimizer_estimate_cardinality
- **One query produces multiple cardinality events**
 - Number of events proportional to complexity of statement
- **Each event produces three major sets of output**
 - InputRelation – Logical operator and part of a parse tree
 - Use to correlate to query plan operators
 - Some operators contain cardinality number
 - Calculator – Info about cardinality calculator types and calculators used
 - StatsCollection – Statistics read for this estimation step and estimates
- **Note: Old cardinality estimator emits no diagnostic events**



86

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Cardinality Calculator Types (Undocumented)

- **DistinctCountCalculator**
- **FilterCalculator**
- **JoinCalculator**
- **ResidualCalculator**
- **SpecialCalculator**



87

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Cardinality Calculators (Undocumented)

- | | |
|--|---------------------------------------|
| ▪ CDVCPanCollapse | ▪ CSeCalcFilteredStatsWrapper |
| ▪ CDVCPanFilter | ▪ CSeCalcFilterWithResidualGuess |
| ▪ CDVCPanJoin | ▪ CSeCalcFixedFilter |
| ▪ CDVCPanLeaf | ▪ CSeCalcFixedJoin |
| ▪ CDVCPanProjectOrGroupBy | ▪ CSeCalcFunctionComparisons |
| ▪ CDVCPanRedundantOJ | ▪ CSeCalcGuessComparisonJoin |
| ▪ CDVCPanRemap | ▪ CSeCalcGuessInList |
| ▪ CDVCPanSplit | ▪ CSeCalcHistogramComparison |
| ▪ CDVCPanTrivial | ▪ CSeCalcIndependentJoin |
| ▪ CDVCPanUnionAll | ▪ CSeCalcMultiColumnEqualsMultiColumn |
| ▪ CDVCPanUniqueKey | ▪ CSeCalcMultiColumnSemiJoin |
| ▪ CDVCPanUpdate | ▪ CSeCalcNegativeFilter |
| ▪ CSeCalcAscendingKeyFilter | ▪ CSeCalcNegativeJoin |
| ▪ CSeCalcAscendingKeyJoin | ▪ CSeCalcOneSided |
| ▪ CSeCalcBitmapFilter | ▪ CSeCalcPointPredsFreqBased |
| ▪ CSeCalcCollnItvlWithFilteredStat | ▪ CSeCalcPointPredWithFilteredStat |
| ▪ CSeCalcColumnInInterval | ▪ CSeCalcPureHiddenSide |
| ▪ CSeCalcCombineFilters_ExponentialBackoff | ▪ CSeCalcSimpleJoin |
| ▪ CSeCalcCombineFilters_Independence | ▪ CSeCalcSimpleJoinWithDistinctCounts |
| ▪ CSeCalcCombineFilters_MinSelectivity | ▪ CSeCalcTrieBased |
| ▪ CSeCalcExpressionComparedToExpression | ▪ CSeCalcTwoStepConjunction |
| | ▪ CSeCalcUniqueKeyFilter |



88

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Cardinality Calculators (Undocumented)

- CDVCPanCollapse
- CDVCPanFilter
- CDVCPanJoin
- CDVCPanLeaf
- CDVCPanProjectOrGroupBy
- CDVCPanRedundantOJ
- CDVCPanRemap
- CDVCPanSplit
- CDVCPanTrivial
- CDVCPanUnionAll
- CDVCPanUniqueKey
- CDVCPanUpdate
- CSeCalcAscendingKeyFilter
- CSeCalcAscendingKeyJoin
- CSeCalcBitmapFilter
- CSeCalcCollIntvlWithFilteredStat
- CSeCalcColumnInInterval
- CSeCalcCombineFilters_ExponentialBackoff
- CSeCalcCombineFilters_Independence
- CSeCalcCombineFilters_MinSelectivity
- CSeCalcExpressionComparedToExpression
- CSeCalcFilteredStatsWrapper
- CSeCalcFilterWithResidualGuess
- CSeCalcFixedFilter
- CSeCalcFixedJoin
- CSeCalcFunctionComparisons
- CSeCalcGuessComparisonJoin
- CSeCalcGuessInList
- CSeCalcHistogramComparison
- CSeCalcIndependentJoin
- CSeCalcMultiColumnEqualsMultiColumn
- CSeCalcMultiColumnSemiJoin
- CSeCalcNegativeFilter
- CSeCalcNegativeJoin
- CSeCalcOneSided
- CSeCalcPointPredsFreqBased
- CSeCalcPointPredWithFilteredStat
- CSeCalcPureHiddenSide
- CSeCalcSimpleJoin
- CSeCalcSimpleJoinWithDistinctCounts
- CSeCalcTrieBased
- CSeCalcTwoStepConjunction
- CSeCalcUniqueKeyFilter



89

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Cardinality Statistics Collections(Undocumented)

- CStCollBaseTable
- CStCollBlackBox
- CStCollCollapse
- CStCollConstTable
- CStCollFilter
- CStCollFilteredStatsShim
- CStCollFudge
- CStCollGroupBy
- CStCollIntervalFilterPlanned
- CStCollJoin
- CStCollJoinExt
- CStCollOrderedTop
- CStCollOuterJoin
- CStCollProject
- CStCollRemap
- CStCollRemoteTable
- CStCollSingleColumnAntiSemiJoin
- CStCollSplit
- CStCollSTVF
- CStCollUnionAll
- CStCollUpdate



90

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Cardinality Estimates and Query Plan

- **When query_optimizer_estimate_cardinality is on**
 - Each trace record contains a stats_collection_id property
- **Query plans iterators contain StatsCollectionId property**
- **These can be correlated**
 - Which trace records refer to each iterator
 - Not every iterator has corresponding trace records



91

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Summary

- **Accurate cardinality estimation critical to good query plan**
- **New Cardinality Estimator introduced in SQL Server 2014**
 - Chooses calculator to use
 - Uses calculator and statistics to calculate cardinality
- **Refinements to basic assumptions**
- **You can choose new or old estimator**



92

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

References

- **Testing Cardinality Estimation Models in SQL Server**
 - <http://dl.acm.org/citation.cfm?id=2304526>
- **Ian Jose's Blog**
 - <http://blogs.msdn.com/b/ianjo/>
- **My Blog**
 - <http://www.sqlskills.com/blogs/bobb/>
- **Joe Sack's Blog**
 - <http://www.sqlskills.com/blogs/joe/>
- **A First Look at the New SQL Server Cardinality Estimator, Ben Navarez**
 - On SQLPerformance.com



93

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

SQLIntersection PRECON02

What's New in SQL Server 2014 Additional Features

Bob Beauchemin
bobb@SQLskills.com



Outline

- Performance Features
- Reliability and Security Features
- Hybrid Cloud Features



95

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Delayed Durability

- Transaction outcome is not returned to caller until after transaction log records are written
- This can slow down any transaction, especially
 - In-memory OLTP (it's the biggest bottleneck)
 - When SQL Server transaction log on Windows Azure storage
- Delayed durability is about loosening transaction guarantee
 - Control return to user before transaction log committed
 - Can lose work on SQL Server or OS crash
- Can be specified
 - Per-database
 - On COMMIT TRANSACTION statement
 - On OLTP compiled stored procedure atomic block



96

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Partitioned Statistics

- **Statistics on partitioned tables are table-level**
 - Recorded at a table-level
 - Updated at a table-level
- **Often, only one partition is the “active” partition**
 - Other partitions contain historical data
- **In SQL Server 2014**
 - Statistics can be created per-partition
 - INCREMENTAL = YES
 - Auto-created partitioned statistics can be created per-database
 - Individual partition's statistics can be updated
 - UPDATE STATISTICS RESAMPLE (ON PARTITIONS)
 - Update can recreate statistics as partitioned
 - INCREMENTAL = YES



97

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Partitioned Stats Statements Affected

- **Partitioned statistics affect DDL**
 - CREATE STATISTICS
 - UPDATE STATISTICS
 - sp_createstats
 - CREATE INDEX
 - ALTER INDEX
 - ALTER DATABASE SET options
 - DATABASEPROPERTYEX
- **Metadata**
 - sys.databases
 - sys.stats



98

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Partitioned Online Index Rebuild

- **Pre-2014**
 - Online index rebuild works only on all partitions
 - Long-running job to rebuild historical partitions
 - Offline index rebuild can rebuild one partition
 - Shorter job
 - “Current” partition is offline during entire job
- **In SQL Server 2014**
 - Online index rebuild works per-partition



99

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Parallel SELECT INTO

- **No code changes needed**
 - Although insert volume must benefit from parallelism
- **Requires Compatibility Level 110 or greater**
- **No parallelism on INSERT... SELECT or INSERT... EXECUTE**



100

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Low Priority Waiting

- **Used in index rebuild operations**
 - Short Sch-M must be taken during online index rebuild
 - Partition switching must take Sch-M on both partitions
 - Sch-M lock is incompatible with shared locks (all readers)
- **If rebuild uses WAIT_AT_LOW_PRIORITY**
 - Waiters for shared locks satisfied before low-priority Sch-M
- **Can specify duration to wait in minutes**
- **Can specify what happens after duration timer expires**
 - ABORT_AFTER_WAIT = [NONE | SELF | BLOCKERS]
 - SELF - Low priority tasks exits with error
 - BLOCKERS – Blocks killed (kill command)
 - NONE - Continue waiting
 - BLOCKERS option requires ALTER ANY CONNECTION privilege



101

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Buffer Pool SSD Extension

- **Performance improved by allocating more memory to buffer pool**
 - Buy and install more memory
 - Use SSD storage to increase buffer pool
- **Since memory usually cheaper than SSD**
 - Use SSD extension on older machines where more memory not an option
 - When you run out of memory slots
- **SSD used as “paged overflow” to store clean pages only**
- **Configured with ALTER SERVER CONFIGURATION statement**
- **Some limitations**
 - Extension must be less than max server memory
 - To increase “SSD memory” – must drop and re-alter
 - To reduce “SSD memory” – must drop, start/restart SQL Server, then alter



102

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Resource Governor Extensions

- **Resource Governor introduced in SQL Server 2008**
 - Meant to guard against runaway tasks that use all resources
 - Goal is more predictable throughput
 - Uses workload groups to define level of service
 - Workload groups are assigned to resource pools
 - Logins assign to workload group by classifier function
- **Pre-SQL Server 2014, you could configure (resource pools)**
 - CPU percent – MAX/MIN
 - Memory percent – MAX/MIN
 - Scheduler Affinity – NUMA or AUTO
 - Addition parameters on workload groups
- **In SQL Server 2014**
 - MAX_OUTSTANDING_IO_PER_VOLUME – global option for IO operations
 - IOPS_PER_VOLUME – MAX/MIN per resource group



103

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

New Diagnostics

- **Performance information available while query executes**
 - sys.dm_exec_query_profiles
- **New features all are instrumented through**
 - Extended events – all
 - Performance Counters – new counters and extension to existing
 - Messages to errorlog – Low priority waiting
 - New DMVs and new columns in existing DMVs



104

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Always On Enhancements

- **Availability Groups**
 - Maximum number of secondary replicas is increased from 4 to 8.
 - Readable secondary replicas now remain available when disconnected from primary
 - Easier setup and replica adding
 - More diagnostics through extended events
- **Failover cluster instances (FCI)**
 - Can use shared cluster volumes (Windows Server 2012) as shared disks



105

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Security

- **New granular permissions**
 - Makes principal of least privilege easier to implement
 - CONNECT ANY DATABASE Permission
 - IMPERSONATE ANY LOGIN Permission
 - SELECT ALL USER SECURABLES Permission
- **Encrypted backups without TDE**



106

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server Configurations

- **There are four main ways SQL Server runs**
 - SQL Server on hardware
 - SQL Server virtualized on premises
 - SQL Server on cloud VM (IaaS)
 - Windows Azure SQL Database (PaaS)
- **Four main differences**
 - Resource Capacity
 - Latency
 - Hardware
 - WASD and Virtual - problems cannot be solved with hardware
 - Instead solve problems with programming, scale out, queuing
 - Throttling



107

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Hybrid Solutions

- **Hybrids introduced in SQL Server 2014**
 - SQL Server in Azure VM as Availability Group Readable Secondary
 - SQL Server databases on Windows Azure Storage
 - SQL Server backup and restore to Windows Azure Storage
 - Includes encryption
 - Automated backup (a.k.a. Smart Backup)
 - Not available for WASD
 - Automated backup available on pre-SQL Server 2014 versions
 - Backup to Azure storage tool
- **Windows Azure storage can be geo-replicated**
 - SQL Server databases on Windows Azure Storage does not support this



108

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Summary

- **Incremental changes**
 - Reduce contention during long jobs
 - Better granularity for partitioned tables
 - Allow resource governor to manage IO limits
 - Allow loosening transaction durability for speed
- **Enhancements for HADR**
- **Incremental security changes**
- **Enable hybrid on-premises and cloud solutions**



109

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Questions?



Don't forget to complete an online evaluation on EventBoard!

Session: PRECON02

What's New In SQL Server 2014

Your evaluation helps organizers build better conferences,
and helps speakers improve their sessions.



Thank you!

