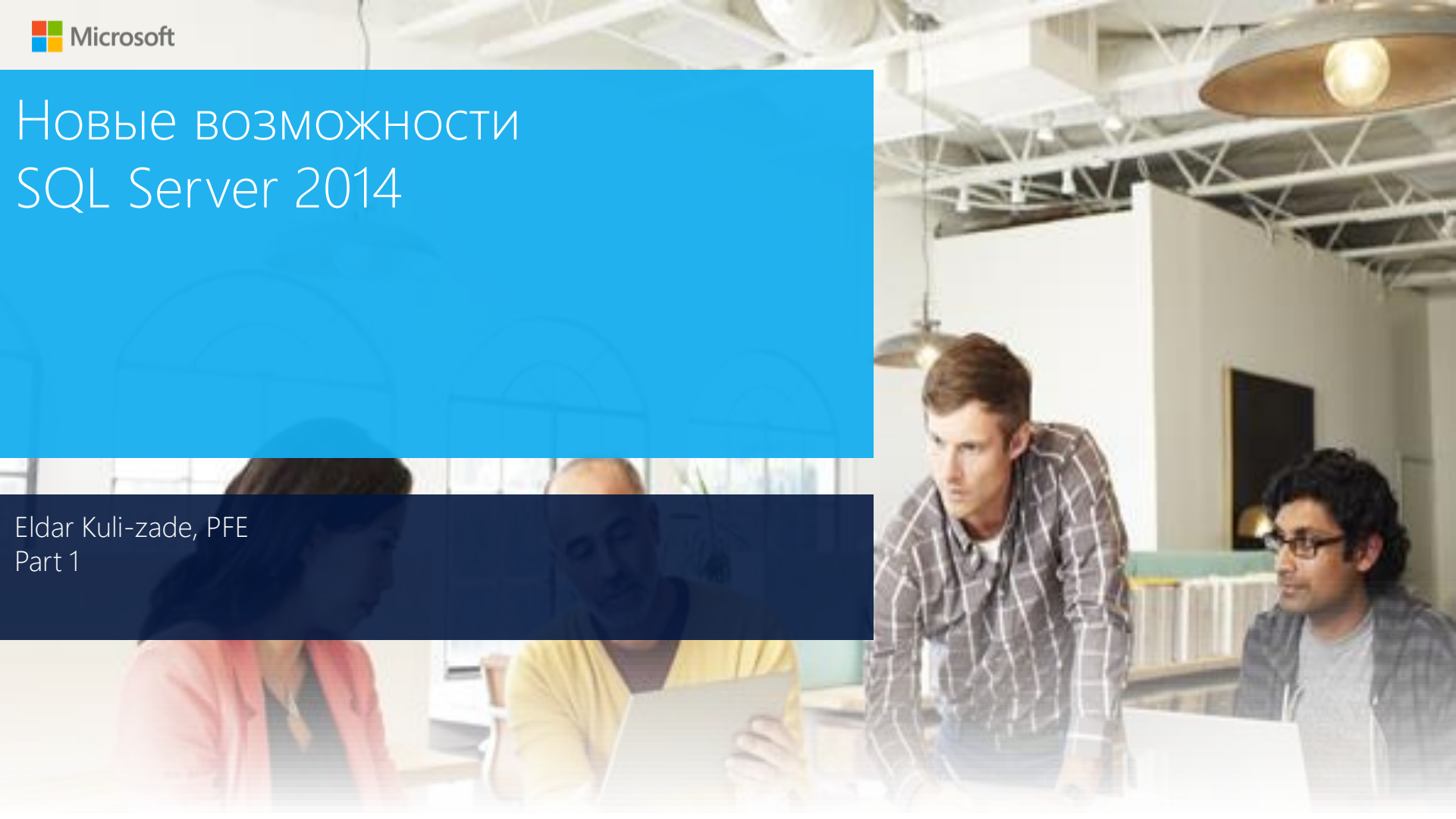


НОВЫЕ ВОЗМОЖНОСТИ SQL Server 2014

Eldar Kuli-zade, PFE
Part 1



New features in components

- Database Engine
- Analytic and Business intelligence
- Installation

Введение:
Пара слов о
себе

- Кули-заде Эльдар
- Работаю с MS SQL Server с 2008 года
- Premier Field Engineer в Microsoft с 2013 года
- Направления SQL Server: производительность, отказоустойчивость

Введение в семинар

- Тема
- Продолжительность
- Структура
- Вопросы/ответы

Что нового в Database Engine

- In-Memory Database
- Backup and Restore
- New Design for Cardinality Estimation
- Delayed Durability
- AlwaysOn Enhancements
- Columnstore compression
- Managing the Lock Priority of Online Operations
- Columnstore Indexes
- Buffer Pool Extension
- Incremental Statistics
- Resource Governor Enhancements for Physical IO Control
- System View Enhancements
- Database Compatibility Level

In-Memory OLTP позволяет добиться значительного повышения производительности и масштабируемости с помощью:

- оптимизированных для доступа к данным, хранимым в памяти;
- управления оптимистичным параллелизмом, устраняющим логические блокировки;
- объектов без блокировки, которые используются для получения всех данных. Потоки, выполняющие транзакционную работу, не используют блокировки или кратковременные блокировки для управления параллелизмом.
- хранимых процедур, скомпилированных в коде языка "C", что на порядок сокращает время выполнения кода.

In-Memory Database - Пример

Пример вставки 10 000 строк в различные виды таблиц при
различном количестве клиентских соединений.

# connections	Hekaton table (elapse time in sec)	SQL Table with Clustered Index (elapses time in sec)	SQL Table with non- clustered index (elapse time in sec)
1	0.420000	0.770000	0.770000
2	0.450000	0.810000	0.820000
5	0.660000	1.950000	2.000000
10	0.740000	10.100000	9.720000
15	0.770000	14.470000	16.590000
20	0.840000	22.520000	23.490000
50	1.650000	72.890000	71.970000
80	2.390000	128.650000	126.990000
100	2.890000	164.560000	161.320000
150	4.320000	268.540000	272.450000
180	5.140000	349.270000	346.080000
200	5.700000	393.960000	391.390000
250	7.180000	523.380000	513.350000

In-Memory Database – Термины и определения

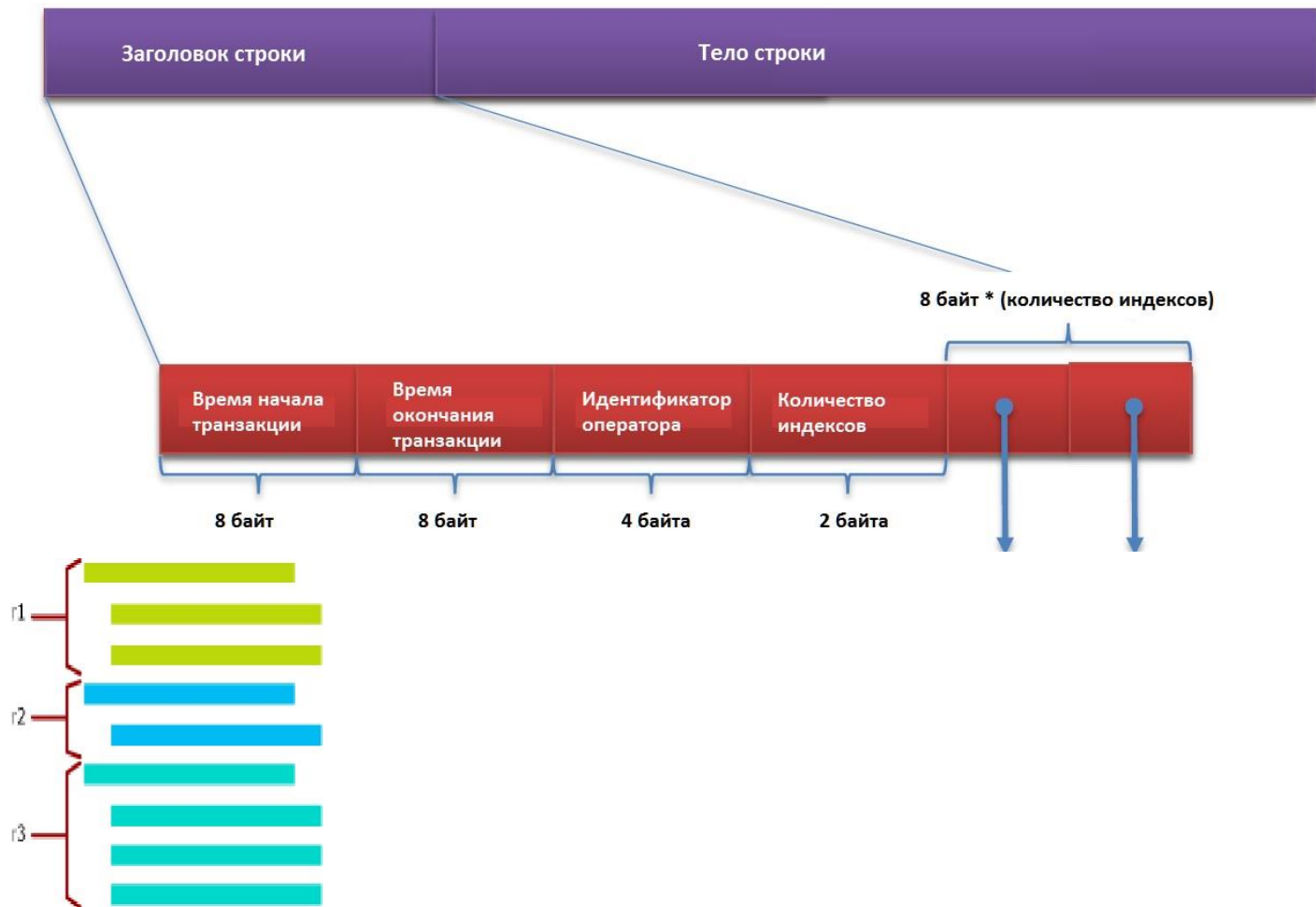
- "In-Memory OLTP database" - OLTP база данных оптимизированная для хранения и обработки в оперативной памяти.
- "Disk based OLTP database" - классическая OLTP база данных оптимизированная для хранения и обработки данных с диска.
- "Memory optimized tables" это таблицы In-Memory OLTP базы данных, которые оптимизированы для доступа при их полном хранении в оперативной памяти.
- "Disk based tables" - таблицы классической базы данных, хранящиеся на диске в виде страниц.
- "Natively Compiled Storage Procedure" - хранимые процедуры код которых компилируется в объектный код (бинарный) и не изменяется никогда от момента создания до удаления.

In-Memory Database – Research

- <http://research.microsoft.com/apps/pubs/default.aspx?id=193594>
- <http://research.microsoft.com/apps/pubs/default.aspx?id=176690>
- <http://research.microsoft.com/apps/pubs/default.aspx?id=178758>
- <http://research.microsoft.com/apps/pubs/default.aspx?id=156645>
- http://research.microsoft.com/en-us/um/people/justinle/papers/ICDE2013_bwtree.pptx
- <http://research.microsoft.com/apps/pubs/default.aspx?id=218305>

- Memory-optimized tables
- Hash-indexes
- Bw-tree indexes
- Native compiled procedures

Memory optimized table (1)



Memory optimized table (2)

- Строки хранятся в “куче”. Рядом могут размещаться строки принадлежащие разным таблицам.
- Максимальное количество hash-индексов на таблице 8.
- “Видимость” строки для транзакция определяется полями “Время начала транзакции” и “Время конца транзакции”
- Поле “Идентификатор оператора” содержит hash оператора внутри блока (batch).
- Таблица обязательно должна содержать хотя бы один индекс.
- Если таблица создается с опцией `DURABILITY=SCHEMA_AND_DATA` она обязательно должна содержать PRIMARY KEY.
- Если таблица создается с опцией `DURABILITY=SCHEMA_ONLY`, то она обязательно должна содержать хотя бы один индекс.
- Структура таблицы компилируется в *.dll.

Memory optimized table (2)

С точки зрения физической организации на диске In-Memory база состоит из файлов трех видов:

- Data - файлы. Файлы в которых хранятся все строки, как те которые реально есть, так и те, которые были подвергнуты обновлению или удалению.
- Delta - файл. Файл хранящий информацию о удаленных строках в Data-файлах.
- Transaction Log - файла, это тот же журнал транзакций обычной (Disk based) базы данных.
- Пара Data и Delta файлов называется Checkpoint File Pair (CFP).

	Data File Name	file_type_desc	Delta file name	file_type_desc
1	00000021-00000171-0003	DATA	00000021-00000178-0002	DELTA
2	00000021-0000017d-0003	DATA	00000021-00000184-0002	DELTA
3	00000021-00000165-0003	DATA	00000021-0000016c-0002	DELTA
4	00000084-00000b8d-0003	DATA	00000084-00000b93-0002	DELTA
5	00000082-000015d0-0002	DATA	00000082-000015d8-0002	DELTA
6	00000084-000009a1-0002	DATA	00000084-000009bf-0002	DELTA
7	00000084-00000936-0002	DATA	00000084-0000093c-0002	DELTA
8	00000084-00000a5e-0003	DATA	00000084-00000a63-0002	DELTA
9	00000084-00000a68-0003	DATA	00000084-00000a6f-0002	DELTA
10	00000021-000000ba-0004	DATA	00000021-00000151-0002	DELTA
11	00000084-00000b98-0003	DATA	00000084-00000b9f-0002	DELTA
12	00000021-00000157-0003	DATA	00000021-00000160-0002	DELTA

Name	Date modified	Type	Size
00000084-00000948-0002	2/4/2014 4:36 AM	File	0 KB
00000084-00000941-0003	2/4/2014 4:36 AM	File	131,072 KB
00000084-00000936-0002	2/4/2014 4:36 AM	File	131,072 KB
00000084-0000093c-0002	2/5/2014 12:07 PM	File	0 KB
00000084-000009bf-0002	2/5/2014 8:04 AM	File	0 KB
00000084-000009a1-0002	2/5/2014 8:04 AM	File	131,072 KB
00000084-00000a68-0003	2/5/2014 12:07 PM	File	131,072 KB
00000084-00000a63-0002	2/5/2014 12:07 PM	File	0 KB

Memory optimized table (3)

- По мере роста количества удаленных строк система производит автоматическое слияние (Merge) файлов.
- Алгоритм слияния основан на наличии свободного места в файлах данных (Data).
- Периодичность выполнения процедуры слияния определяется внутренними политиками сервера.
- Если в Data-файле удалены все строки, то такая пара файлов переводится в состояние TOMBSTONE и со временем удаляется внутренним процессом сбора мусора (Garbage Collection).
- Максимальное количество файлов (Data и Delta) 8192.
- Идеальными размерами файлов для компьютеров с объемом RAM менее 16 ГБ являются Data-файлы 16 МБ и Delta-файлы 1 МБ.
- Идеальными размерами файлов для компьютеров с объемом RAM более 16 ГБ являются Data-файлы 128 МБ и Delta-файлы 8 МБ.
- Размер файлов, как правило, в два раза больше, чем размер данных в памяти.

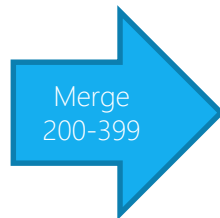
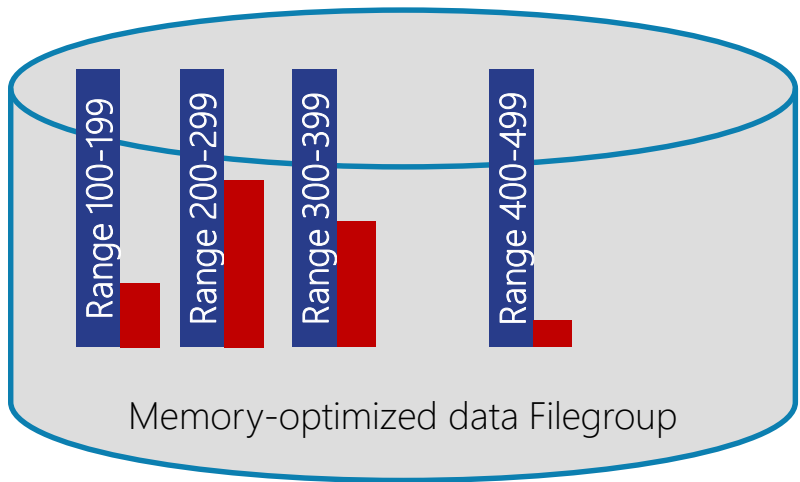
Memory
optimized table
– Политики
слияния
файлов

SQL Server 2014 использует следующие политики слияния:

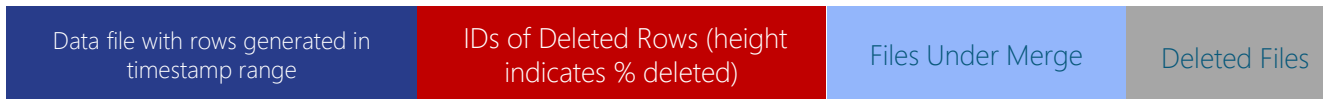
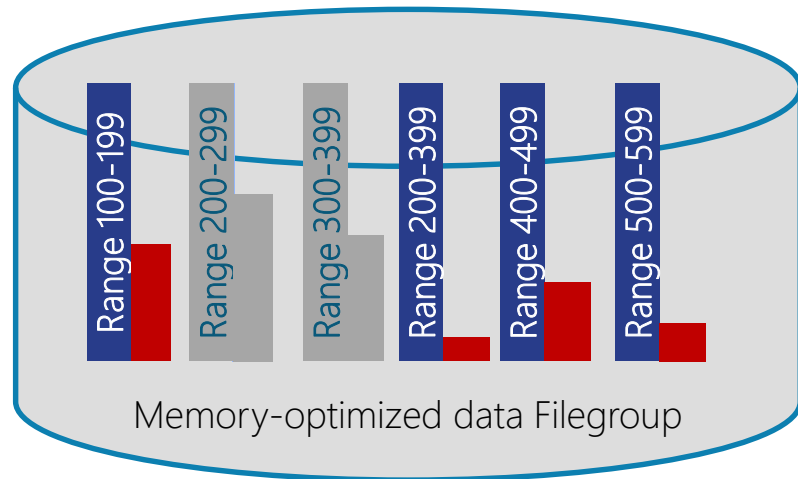
- Операция слияния планируется для выполнения если два и более последовательных файла могут быть объединены в один, таким образом, что результирующий файл будет соответствовать идеальным размерам файла.
- Самослияние срабатывает если размер файла превышает 256 МБ и в файле половина строк числится удаленными. Тогда происходит уменьшение размеров файла до идеального.
- Существует возможность ручного слияния с помощью процедуры `sys.sp_xtp_merge_checkpoint_files`.
- Флаг трассировки 9851 для отключения автоматического слияния файлов (для тестирования).

Merge

Files as of Time 500



Files as of Time 600



Memory
optimized table
– Жизненный
цикл файлов (1)

CFP файлы могут находиться в следующих состояниях:

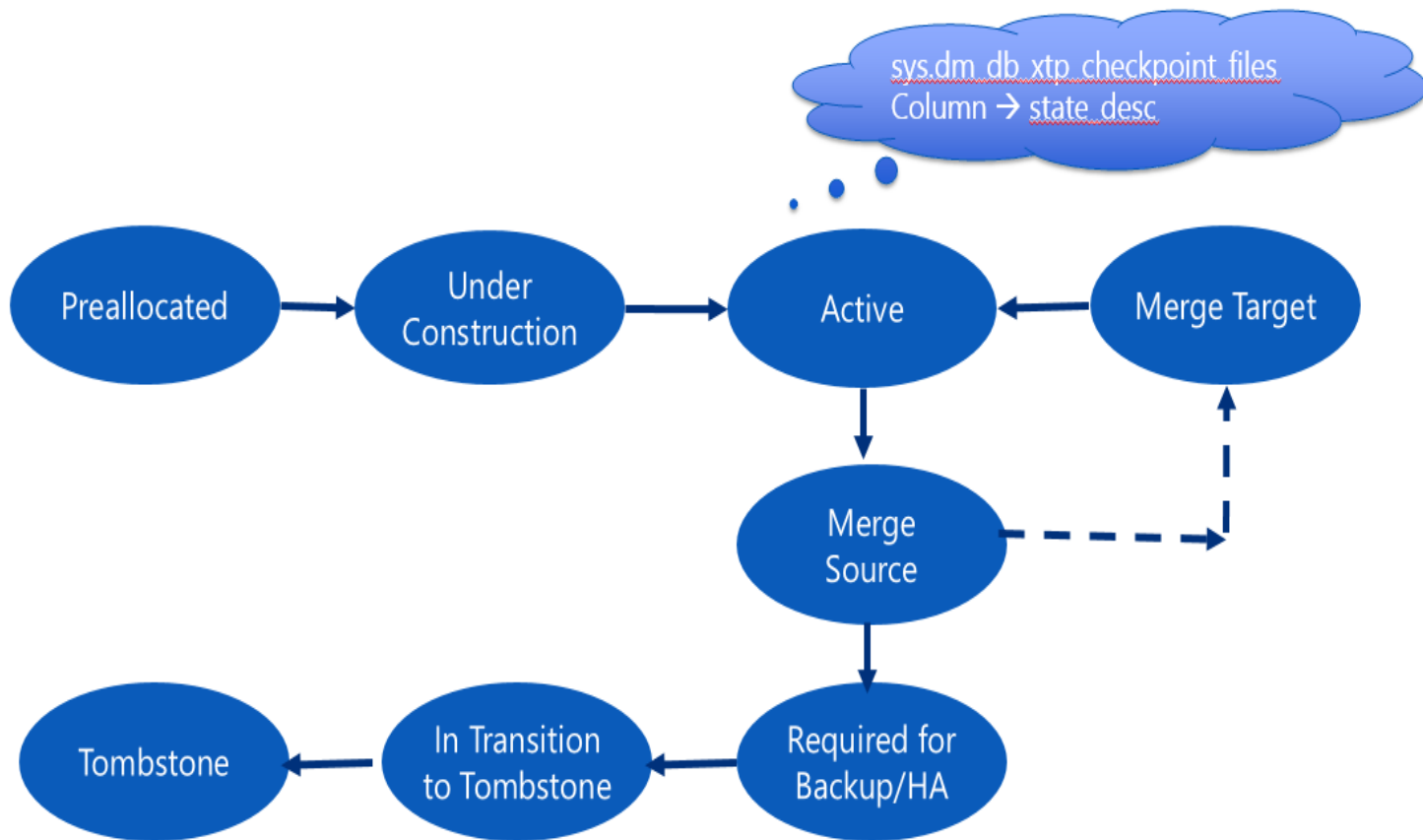
- PRECREATED – небольшое количество предварительно созданных полноразмерных файлов. Количество создаваемых файлов соответствует количеству процессоров, но не менее 8-ми. Файлы создаются, чтобы уменьшить временные задержки, которые могут возникнуть при их срочном создании.
- UNDER CONSTRUCTION – файлы, в которых еще не отображены изменения, сделанные в журнале транзакций, поскольку Checkpoint-процесс не выполнялся для этих изменений.
- ACTIVE – содержат строки соответствующие всем предыдущим закрытым Checkpoint. Расчетный размер этих файлов в два раза больше, чем размер соответствующих объектов в оперативной памяти.
- MERGE TARGET – файл будет содержать консолидируемые строки из объединяемых файлов в соответствии с политикой слияния. После окончания слияния файл перейдет в состояние ACTIVE.

Memory
optimized table
– Жизненный
цикл файлов
(2)

CFP файлы могут находится в следующих состояниях:

- MERGED SOURCE – файл будет содержать консолидируемые строки, которые будут копироваться из него в файл получатель (MERGE TARGET).
- REQUIRED FOR BACKUP/HA – в это состояние переходят файлы которые были в состоянии MERGED SOURCE. Они находятся в этом состоянии до выполнения резервной копии. После выполнения резервной копии файлы из этого состояния могут перейти в состояние IN TRANSITION TO TOMBSTONE, TOMBSTONE и далее вычищены процессом сбора мусора.
- IN TRANSITION TO TOMBSTONE – ожидание потока, который переведет файлы в состояние очистки сборщиком мусора.
- TOMBSTONE - ожидание очистки процессом сбора мусора (Filestream Garbage Collection).

Memory
optimized table –
Жизненный
цикл файлов (3)



Memory
optimized table
– Демонстрация
Memory
optimized
table.sql

- Создание базы.
- Создание таблицы.
- Просмотр состояния файлов.

Hash-indexes (1)

- Хеширование, что и зачем?
- Требования к хешированию.
- Алгоритмы
 - MD5, SHA1, SHA256, SHA384 и пр.
 - ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012.
- Hash-индексы
 - Не присутствуют на диске, а строятся в момент загрузки данных.
 - Используют одну и ту же детерминистическую функцию
 - В основе лежит hash-таблица (hash-bucket table), содержащая ссылки на строки.
 - Количество hash-таблиц зависит от количества hash-индексов
 - Размер hash-таблицы должен задаваться при создании индекса, исходя из предполагаемого количества уникальных значений в наборе данных.
 - Возможны hash-коллизии.

Hash-indexes (2)

- Размер hash-таблицы указывается при создании Memory-optimized таблицы.
- Размер hash-таблицы указывается в количестве значений hash-ключей.
- Размер одной записи hash-таблицы 8 байт.
- Указав чрезмерный размер hash-таблицы (значительно больше чем планируемое количество уникальных значений в таблице) вы резервируете оперативную память, которая не может быть использована и будет потеряна для системы.
- Указав недостаточный размер hash-таблицы, вы получите ситуацию, при которой сервер будет вынужден двигаться по цепочке ссылок, ища нужную запись (большое количество hash-коллизий).

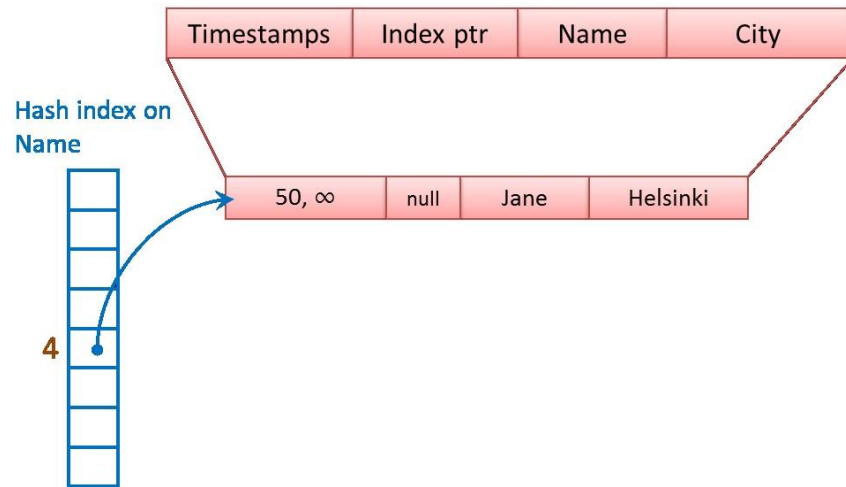
Hash-indexes (3)

- Для балансировки используется Poison-алгоритм.
- Суть которого состоит в том, что если вы пытаетесь распределить N-значений ключей среди M-значений hash-функций, то:
 - приблизительно 1/3 ячеек будет пустой,
 - приблизительно 1/3 ячеек будет содержать по одному значению,
 - приблизительно 1/3 ячеек будет содержать по два значения,
 - малая часть будет содержать два и более значения.

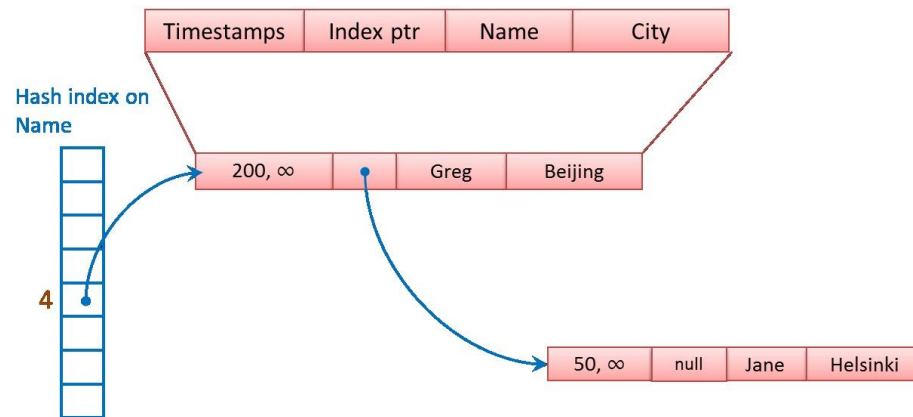
Выводы:

- Даже если указать количество hash-ключей (BUCKET_COUNT) равное количеству уникальных значений, то часть ключей останется свободной, а, следовательно, возникнут цепочки строк.
- Для уменьшения длин цепочек строк необходимо указывать количество BUCKET_COUNT больше, чем количество уникальных значений в два и более раз.

Hash-indexes (4)

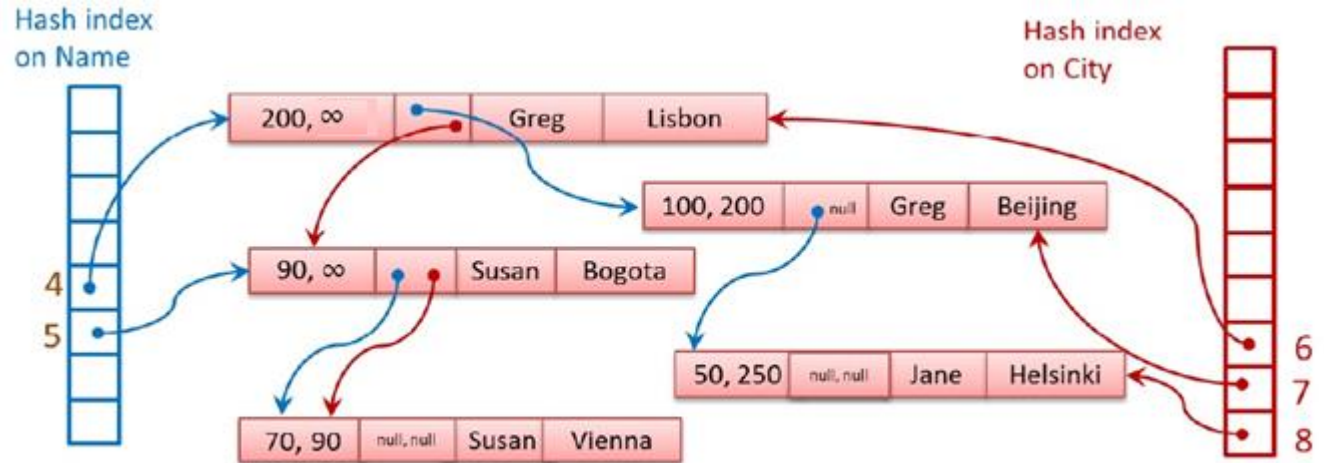


Пример Hash-коллизии



Пример двух Hash-индексов и модификации данных

Hash-indexes (5)



Hash-indexes - Рекомендации

- В большинстве случаев параметр `BUCKET_COUNT` должен быть от 1 до 2 раз больше чем количество уникальных значений в таблице.
- Если соотношение между количеством уникальных значений и общим количеством строк составляет 100 и более, то параметр `BUCKET_COUNT` должен быть 8...10 и более раз больше, чем количество уникальных значений в таблице.

Hash-indexes - Недостатки

Основные недостатки hash-индексов:

- Hash-коллизии
- Невозможность поиска по диапазону значений
- Невозможность поиска по части значения
- Необходимость предвидеть статистику хранимых данных.

In-Memory
Database –
Демонстрация
Hash indexes.sql

- Hash-индексы
- Цепочки
- Коллизии
- Выделение памяти по индексу и таблице

Bw-indexes (1)

Основные недостатки B-tree индексов:

- Наложение блокировок (Lock) и защелок (Latch).
- Необходимость балансировки дерева, а отсюда наложение Latch-ей
- Деление страниц связанной с фиксированным размером страниц и опять же, связанное с этим наложение Latch-ей.

Итак, основная проблема стандартных B-tree индексов – невозможность выполнения модификаций без блокировки данных на логическом (Lock) и физическом (Latch) уровнях.

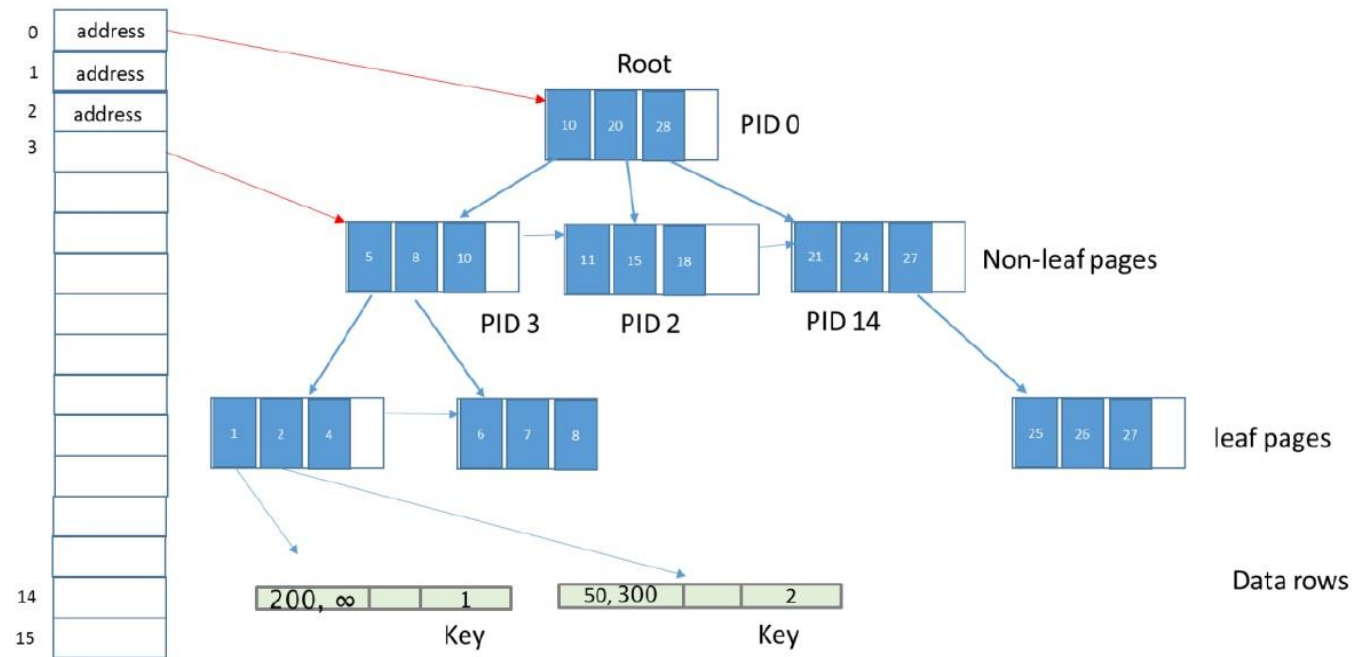
Bw-indexes (2)

Отличия Bw-индексов от B-tree индексов:

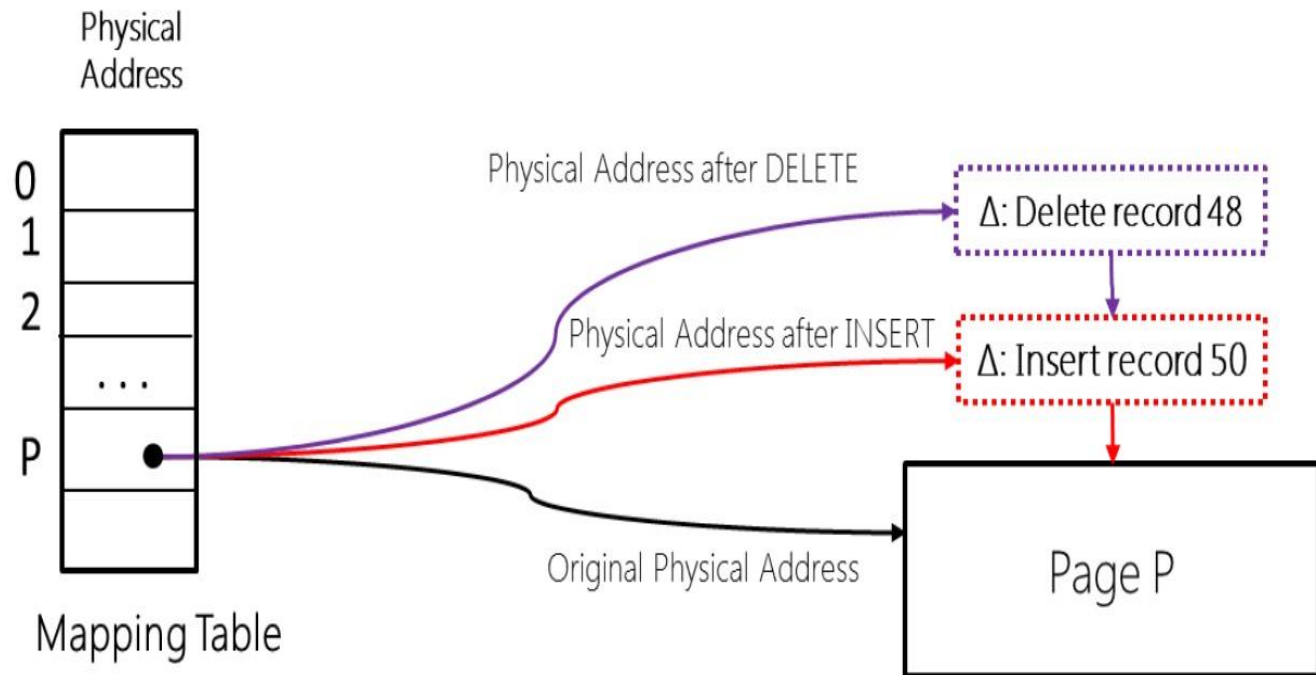
- Номер страницы – это логический номер не связанный с физическим размещением страницы;
- Размеры страниц не фиксированы.
- Страницы листового уровня всегда ссылаются на строки с данными (подобно классическим некластерным индексам).
- Модификация данных всегда выполняется без блокировки, поскольку строка никогда не модифицируется после ее создания.
- Модификация происходит путем создания delta-записи.

Bw-indexes - Структура

Page Mapping Table

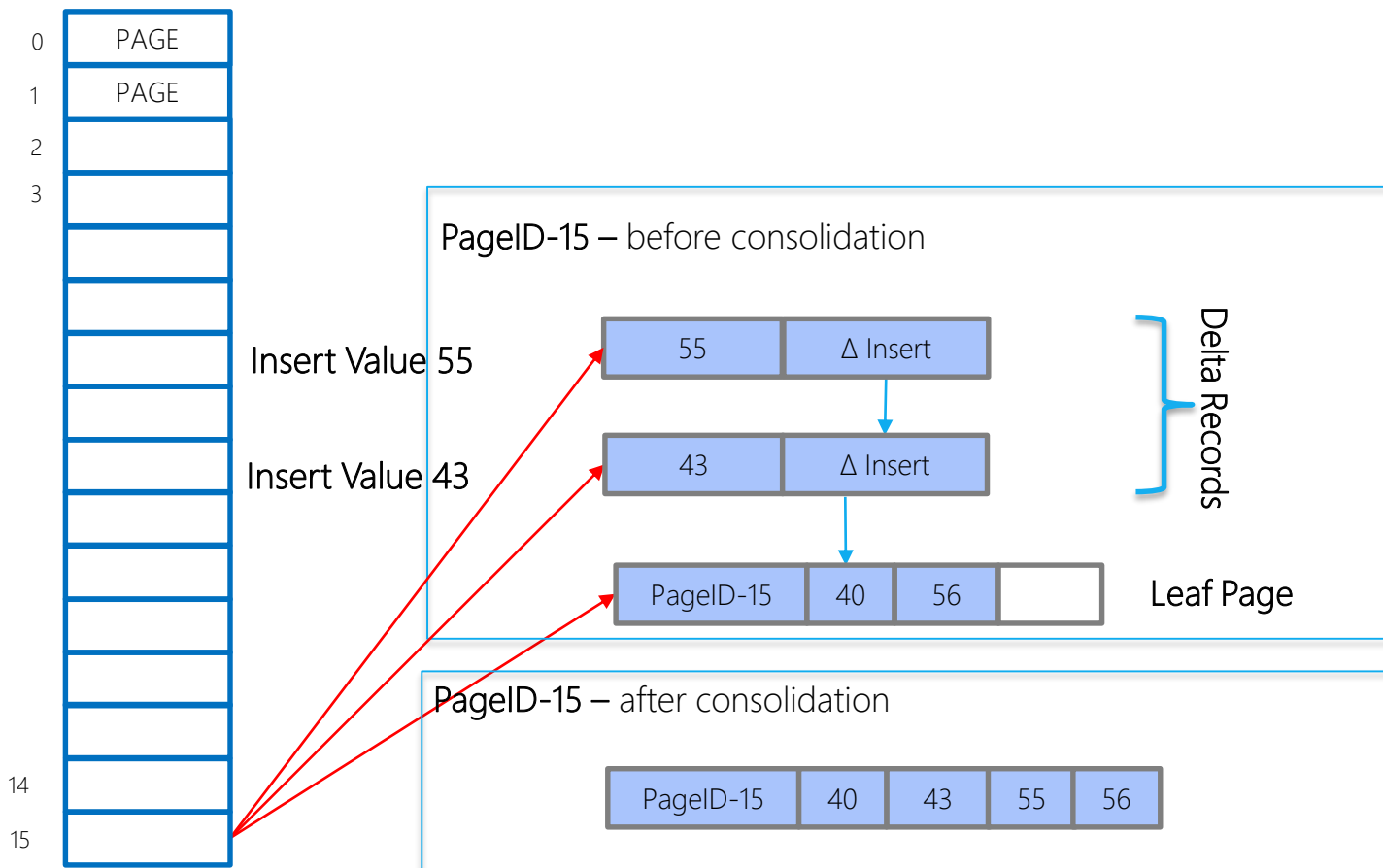


Bw-indexes - Модификация



Bw-indexes - Консолидация

Page Mapping Table



Bw-indexes –
Принятие
решения об
использовании

Operation	Hash index	Nonclustered index (Range)	Disk-based index
Index Scan	Yes	Yes	Yes
Index seek on equality predicate(s) (=)	Yes (По полному ключу)	Yes	Yes
Index seek on inequality (>, <, <=, '>=)	Not (index scan)	Yes	Yes
Sort-order matching the index definition	No	Yes	Yes
Sort-order matching the reverse of the index definition	No	No	Yes

In-Memory
Database –
Демонстрация
Bw-indexes.sql

- Bw-индексы

Access to Memory- optimized table

Доступ к таблицам может быть осуществлен:

- Natively compiled хранимых процедур – наиболее быстрый способ доступа к данным;
- Стандартного T-SQL;

Access to Memory- optimized table (2)

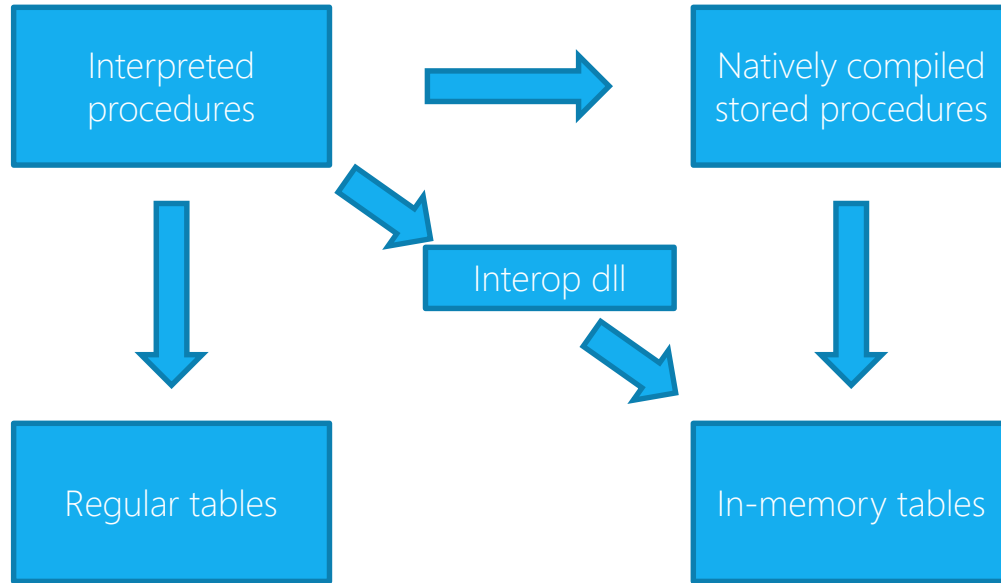
Особенности Natively compiled хранимых процедур:

- Преобразуются с язык "С" и далее в объектный код, и dll;
- Компилируются один раз при создании и далее перекомпилируются при рестарте сервера.
- Результат компиляции (алгоритм и код) зависит от среды существовавшей при компиляции (в том числе от статистики);
- Объекты, на который ссылается такая хранимая процедура, не могут быть изменены DDL операторами.

Access to Memory- optimized table (3)

- Метод доступа через T-SQL еще называется Inter-Op или Interpreted TSQL access;
- Этот метод доступа снимает ряд ограничений которые присутствуют в Natively Compiled хранимых процедурах:
 - Truncate table
 - MERGE используя Memory-optimized table как целевые
 - Dynamic и keyset cursors;
 - Cross database запросы или транзакции;
 - Связанные сервера;
 - Блокировочные подсказки (hints).
- Использование T-SQL для выполнения:
 - Ad hoc запросов и административных задач;
 - Запросов для построения отчетов;
 - Одиночных DML операторов (SELECT, UPDATE, INSERT);
 - Первого шага к миграции из стандартной в In-Memory базу данных.

Access to
Memory-
optimized table
(4)



Исполнение
Native-compiled
хранимых
процедур -
проблемы
(особенности)

- Актуальный план выполнения Native-compiled хранимой процедуры не отображается никакими средствами.
- Статистика выполнения Native-compiled хранимых процедур не выводится через SET STATISTICS IO.
- Не используется параллелизм.
- Используется только алгоритм NESTED LOOP.
- Компиляция Native-compiled хранимой процедуры может выполняться достаточно долго, поэтому, по возможности, не создавайте их динамически (по ходу выполнения).
- DBCC freeproccache и DBCC freesystemcache не могут использоваться для очистки процедурного кэша Native-compiled хранимых процедур.
- Для контроля объемов памяти, используемой для хранения необходимо применять sys.dm_os_memory_object с группировкой по объекту MEMOBJ_XTPPROCACHE.

Пример
синтаксиса
хранимой
процедуры

```
CREATE PROCEDURE pSample
    @parameter int = 0
WITH SCHEMABINDING, EXECUTE AS OWNER, NATIVE_COMPILATION
AS
BEGIN ATOMIC WITH
(
    TRANSACTION ISOLATION LEVEL = SNAPSHOT,
    LANGUAGE = N'us_english'
)
<procedure body>
END
```

owner, self, user

Тело процедуры
всегда в транзакции

SNAPSHOT
REPEATABLE
READ
SERIALIZABLE

Язык сообщений

Обязательно

Мониторинг Native-compiled хранимых процедур

- Статистика выполнения не выводится через SET STATISTICS IO.
- Можно использовать SET STATISTICS TIME.
- sys.dm_exec_query_stats и sys.dm_exec_procedure_stats, по умолчанию не содержат статистики.
- Включение сбора статистики производится путем включения ее сбора через sp_xtp_control_proc_exec_stats и sp_xtp_control_query_exec_stats.
- Выполнение этих процедур не генерирует Xevent-событие sp_statement_starting, а только событие sp_statement_completed.
- Счетчики Performance Monitor с расширением XTP
- Использование Xevent (Все события относятся к каналам "Analytic" и "Debug").

Access to Memory-
optimized table –
Демонстрация –
Access to Memory-
optimized table .sql

Использование T-SQL для выполнения:

- T-SQL хранимых процедур;
- Native Compiled хранимых процедур.

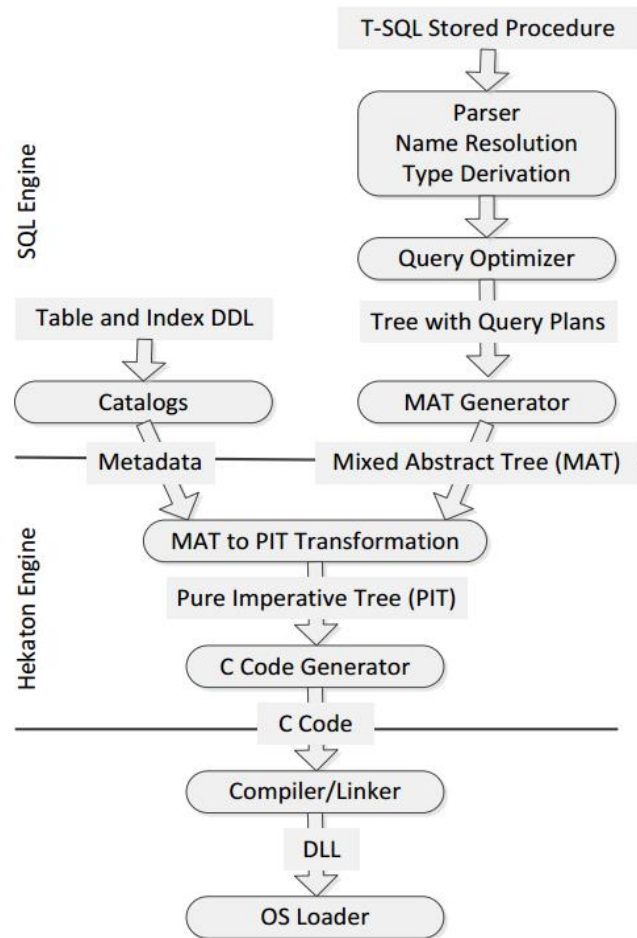
Компиляция объектов – Зачем это надо?

- Все хотят чтобы СУБД работала быстрее
- А зачем такой радикальный подход?
 - Проблема в том, что время исполнения T-SQL кода достаточно равномерно распределено между компонентами
 - Нужно уменьшить «количество кода»
 - Следовательно надо все переписать с нуля



Компиляция объектов – Алгоритм

- В начале создается обычный план исполнения
- Затем он в несколько шагов преобразуется в код на языке C
- После компиляции получается dll, которую подключает процесс SQL Server

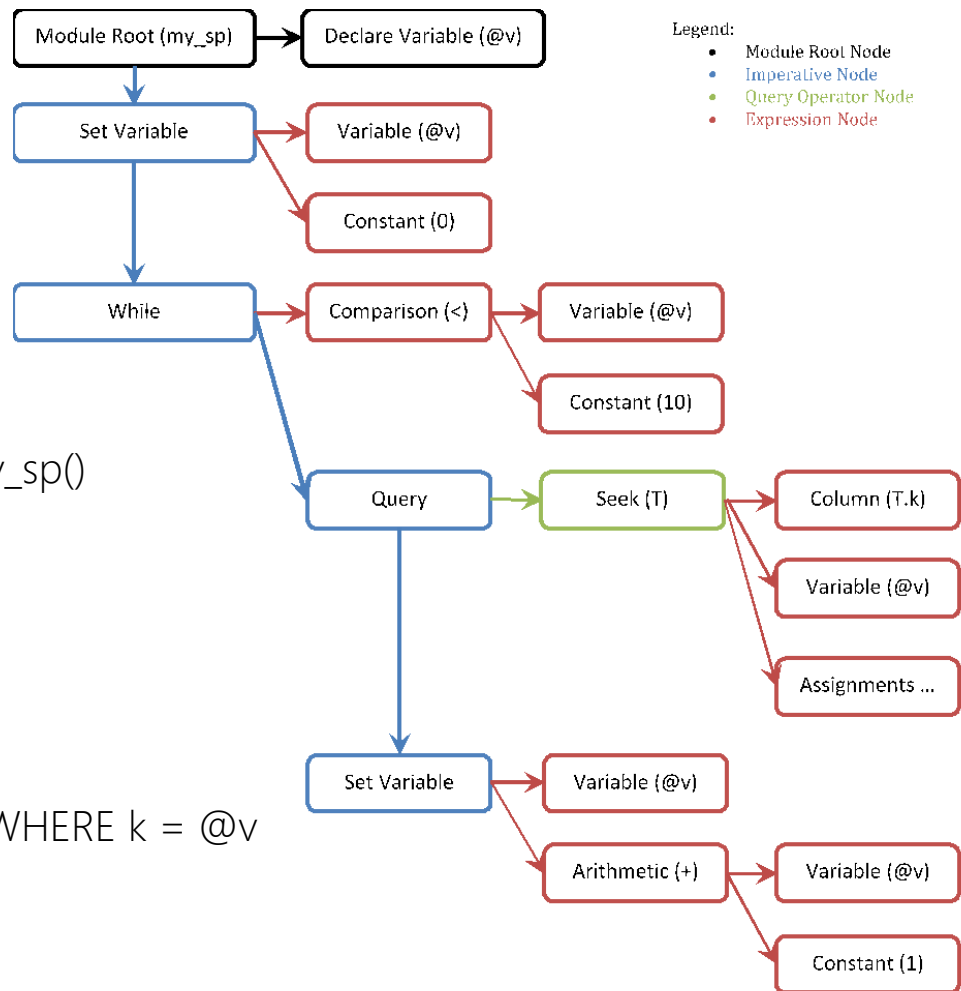


Компиляция объектов – Mixed Abstract Tree (MAT)

- SQL Server использует parser, algebrizer, и query optimizer чтобы преобразовать T-SQL в “Bound tree”.
- “Bound tree” преобразуется в “Mixed abstract tree” (MAT) полностью представляющий хранимую процедуру.
- MAT далее преобразуется в Pure Imperative Tree (PIT).
- PIT служит основой для формирования C-представления типов данных, используемых в T-SQL.
- Используется C-компилятор, входящий в состав SQL Server 2014 (cl.exe) для компиляции и сборки кода в dll.







Компиляция
объектов –
Mixed Abstract
Tree (MAT) -
пример

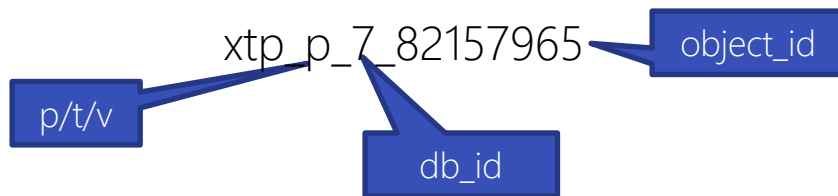
```
CREATE PROCEDURE my_sp()
AS
BEGIN
DECLARE @v INT
SET @v = 0
WHILE @v < 10
BEGIN
    SELECT ... FROM T WHERE k = @v
    SET @v = @v + 1
END
```



Компиляция объектов – Файлы

- Файлы размещаются в папке xtp\<db id>\
- При рестарте SQL Server все файлы удаляются и пересоздаются
- Удаление и пересоздание может занять значительное время.

Name	Date modified	Type	Size
 xtp_p_7_821577965.c	15.05.2014 19:05	C File	10 KB
 xtp_p_7_821577965.dll	15.05.2014 19:05	Application extens...	75 KB
 xtp_p_7_821577965.obj	15.05.2014 19:05	OBJ File	90 KB
 xtp_p_7_821577965.out	15.05.2014 19:05	OUT File	1 KB
 xtp_p_7_821577965.pdb	15.05.2014 19:05	PDB File	587 KB
 xtp_p_7_821577965.xml	15.05.2014 19:05	XML File	17 KB



Компиляция объектов – Ограничения

- Поддерживаются:
 - SELECT/INSERT/UPDATE/DELETE
 - TRY/CATCH/THROW
 - RETURN
 - SET
 - IF and WHILE
 - Некоторые хинты
 - In-memory table types
- Не поддерживаются
 - MERGE
 - временные таблицы
 - табличные переменные
 - Min и Max для текста
 - Не все типы данных
 - Не все функции
 - Не везде можно делать подзапросы

Native Compilation Advisor – ищет неподдерживаемые конструкции

Компиляция объектов – проблемы

- Могут использоваться флаги трассировки, но, к сожалению, они не публичны
- События Xevent
 - xtp_create_procedure,
 - xtp_matgen,
 - xtp_deploy_done.
- Для работы с этими событиями необходимо включить Debug Chanel.
- Ошибка 41312 появляется при невозможности вызвать компилятор cl.exe. Пояснение к ошибке содержится в State.
- Ошибка 41313 появляется если компиляция не может быть успешной. Пояснение к ошибке может быть получено из файла *.out.

Компиляция
объектов –
Демонстрация –
Компиляция
объектов.sql

Использование T-SQL для выполнения:

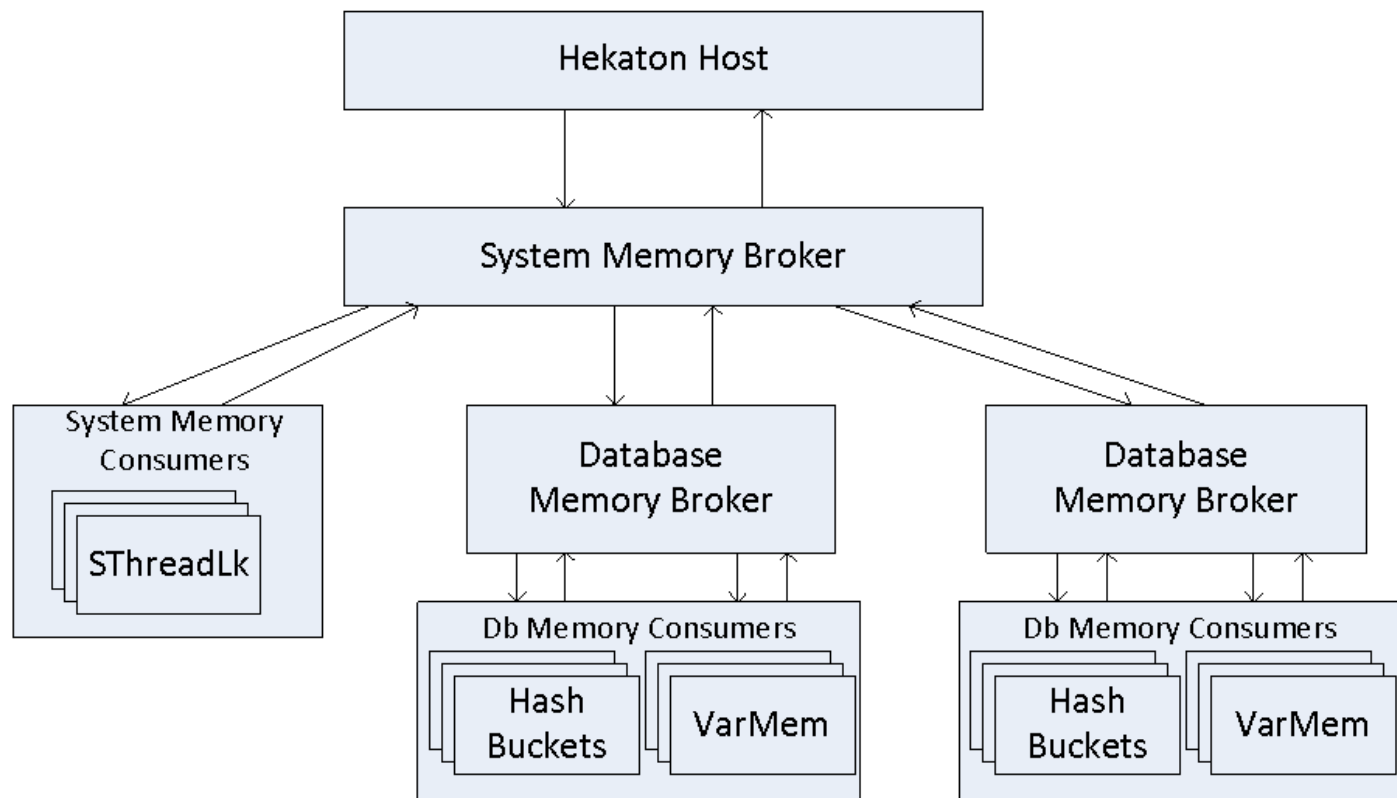
- T-SQL хранимых процедур;
- Native Compiled хранимых процедур.

What's new SQL Server 2014

Eldar Kuli-zade, PFE
Part 2



In-Memory Databases – Memory Management



Memory Management - System Memory Broker

- System Memory Broker – компонент напрямую взаимодействующий с хостом.
- Один Hekaton Memory Broker для каждого NUMA узла.
- Можно выполнить запрос `sys.dm_os_memory_brokers` для `MEMORYBROKER_FOR_XTP`.
- System Memory Broker обеспечивает следующие сервисы
 - Пересылает запросы размещения/удаления памяти к хосту.
 - Отслеживает общесистемных потребителей (system memory consumers) и database memory brokers
 - Передает :“Low memory notification” к System memory consumers и Database memory brokers.
 - Является публичным API для DMVs.

Memory Management – Garbage Collection

- In-Memory Database – это система основанная на мульти версионности данных, что требует наличия механизма удаления не актуальных версий.
- Такой механизм называется Garbage Collection.
- Задача Garbage Collection – удаление старых строк, которые ныне не используются и должны быть вычищены.
- Механизм работает для строк которые:
 - Удалены и должны стать невидимыми.
 - Обновляемые строки, помеченные на удаление поскольку появилась новая версия строки.
 - Строки, которые должны быть удалены вследствие отката (Rollback) транзакции.

Memory Management – Конфигурирование памяти

- Интеграция с Resource Governor.
- Lock Page In Memory
- Max/Min server memory

Memory
Management –
Интеграция с
Resource
Governor.

- Создать Resource pool, ограничив размер памяти, выделяемой для In-Memory Database.
- Связать ресурсный пул с базой данных используя `sys.sp_xtp_bind_db_resource_pool`.
- Установить базу данных в Offline, а затем в Online.
- Если вы хотите отсоединить ресурсный пул от базы данных используйте `sys.sp_xtp_unbind_db_resource_pool`.

Memory Management – Lock Page in Memory

- Если не использовать “Lock Page In Memory”, то при внешнем нажиме на память SQL Server начнет выгружать память содержащую In-Memory Database объекты в файл подкачки.

Memory Management – Max/Min server memory

- Max/Min server memory влияет на распределение всей памяти, включая память используемую In-Memory Database.

Memory
Management –
Демонстрация -
Memory
Management.sql

In-Memory
Database – DMV
и просмотр
метаданных (1)

- Sys.dm_db_index_physical_stats:
- Sp_spaceused & Sp_helpfile:
- Missing index DMVs
- ObjectpropertyEx
- Sys.sql_modules и sys.all_sql_modules
- Sys.tables:
- sys.indexes
- sys.data_spaces

In-Memory
Database – DMV
и просмотр
метаданных (2)

Sys.dm_db_index_physical_stats:

Имеет по одной записи для каждого индекса. Поля фрагментации и количества страниц для них не применимы.

Sp_spaceused & Sp_helpfile:

Не отображают информацию по In-Memory базе данных.

Missing index DMVs

Не применимы.

ObjectpropertyEx

Специальный атрибут TableIsMemoryOptimized.

Sys.sql_modules and sys.all_sql_modules

Столбец uses_native_compilation. Если вы хотите получить информацию по Native Compiled хранимой процедуре необходимо выполнить объединение с таблицей sys.procedures.

In-Memory
Database – DMV
и просмотр
метаданных (3)

sys.tables

Столбцы **Durability, durability_desc, is_memory_optimized**

sys.indexes

Имеет значение 7 для типа индекса и NONCLUSTERED HASH для type_desc.

sys.data_spaces

Имеет тип FX и MEMORY_OPTIMIZED_DATA_FILEGROUP для type_desc.

In-Memory Database – DMV и просмотр метаданных (4)

sys.dm_db_xtp_checkpoint_stats
sys.dm_db_xtp_checkpoint_files
sys.dm_db_xtp_merge_requests

sys.dm_db_xtp_gc_cycle_stats
sys.dm_xtp_gc_queue_stats
sys.dm_xtp_gc_stats

sys.dm_db_xtp_index_stats
sys.dm_db_xtp_hash_index_stats
sys.dm_db_xtp_nonclustered_index_stats

sys.dm_xtp_system_memory_consumers
sys.dm_db_xtp_memory_consumers

sys.dm_db_xtp_transactions
sys.dm_xtp_transaction_stats

sys.dm_db_xtp_object_stats
sys.dm_db_xtp_table_memory_stats

In-Memory Database – Совместимость и ограничения

- Проблемы с некоторыми типами процессоров
- Поддерживаемые типы данных
- Ограничения по использованию программных инструкций
- Административные ограничения
- Интеграция с другими возможностями SQL Server.

Совместимость и ограничения – Типы процессоров

- Некоторые типы AMD процессоров не поддерживают инструкцию CMPXCHG16B, лежащей в основе операции Compare&Swap.
- При этом при создании файловой группы появится ошибка

The model of the processor on the system does not support creating MEMORY_OPTIMIZED_DATA. This error typically occurs with older processors. See SQL Server Books Online for information on supported models.

Совместимость и ограничения – Типы данных

Не поддерживаются типы данных:

- Datetimeoffset
- Varbinary(max)
- Varchar(max)
- Nvarchar(max)
- Xml
- Text
- Image
- Sql_variant
- ROWGUIDCOL

Нельзя использовать:

- DML триггеры
- FOREIGN KEY
- CHECK ограничения
- IDENTITY столбцы только IDENTITY (1, 1)
- UNIQUE индексы (только как PRIMARY KEY)
- MARS
- DTC
- DELETE with FROM clause
- UPDATE with FROM clause
- CTE
- CASE

Смотри [http://msdn.microsoft.com/en-us/library/dn246937\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/dn246937(v=sql.120).aspx)

In-Memory
Database –
Административные
ограничения

Нельзя использовать:

- ALTER TABLE
- ALTER INDEX
- REBUILD INDEX
- AUTO CLOSE
- AUTO UPDATE STATISTICS
- ATTACH_REBUILD_LOG
- DATABASE SNAPSHOT
- DBCC CHECKDB – пропускает
- DBCC CHECKTABLE – выдает ошибку

Совместимость и ограничения –
Интеграция с
другими
возможностями

Возможности	Степень поддержки
Failover Cluster	Полностью
AlwaysOn:	Полностью. Кроме: Non-durable таблицы. Они будут создаваться на вторичных репликах, но они будут пусты.
Replication	Ограниченная поддержка. Нельзя использовать Memory optimized таблицы как статью в публикации (Article in Publication) или как часть Подписчика (Subscriber).
Log shipping	Полностью
Mirroring	Нет

Что нового в
Database
Engine -
Backup and
Restore
Enhancements

Encryption for Backups

Encryption – Overview

- Шифрование помогает обезопасить данные при их хранении и транспортировке.
- Шифрование резервной копии может применяться совместно с TDE.
- Поддерживается несколько доступных алгоритмов, включая AES 256.
- Ключи шифрования могут быть интегрированы с Extended Key Management (EKM) провайдерами.
- Ключ шифрования резервной копии размещен в заголовочной части Backup и зашифрован открытым ключом (Public Key) сертификата или асимметричного ключа.
- Сертификат, точнее его закрытый ключ (Private Key) шифруется мастер-ключом базы данных (Database Master Key).
- Возможно использование сжатия и шифрования. При этом сначала производится сжатие, а потом шифрование резервной копии.

Encryption – Overview

- Шифрование помогает обезопасить данные при их хранении и транспортировке.
- Шифрование резервной копии может применяться совместно с TDE.
- Поддерживается несколько доступных алгоритмов, включая AES 256.
- Ключи шифрования могут быть интегрированы с Extended Key Management (EKM) провайдерами.
- Ключ шифрования резервной копии размещен в заголовочной части Backup и зашифрован открытым ключом (Public Key) сертификата или асимметричного ключа.
- Сертификат, точнее его закрытый ключ (Private Key) шифруется мастер-ключом базы данных (Database Master Key).
- Возможно использование сжатия и шифрования. При этом сначала производится сжатие, а потом шифрование резервной копии.

Backup Encryption – Предварительные требования

- Должен быть предварительно создан Database Master Key в базе данных master (Это асимметричный ключ, который используется для защиты private ключей сертификатов и асимметричных ключей в базе данных).
- Должен быть создан сертификат или асимметричный ключ для шифрования ключей шифрования резервных копий.
- **Примечание: Поддерживаются только асимметричные ключи размещенные в Extended Key Management (EKM).**

Backup Encryption - Ограничения

- Поддерживаются только асимметричные ключи размещенные в Extended Key Management (EKM).
- SQL Server Express и SQL Server Web не поддерживают шифрование резервных копий, однако восстановление зашифрованных резервных копий на этой версии поддерживается.
- Предыдущие версии SQL Server не поддерживают эту возможность.
- Присоединение резервной копии к существующему backup set не поддерживается для зашифрованных резервных копий.

Backup Encryption - Методы

- SQL Server Management Studio
- Transact-SQL

Backup Encryption – Рекомендации по применению

- Создайте резервную копию сертификата и Database Master Key на другой компьютер.
- Восстановление зашифрованной резервной копии возможно только с использованием сертификата и ключа с помощью которых резервная копия была защищена. Обновление сертификата на сервере, при отсутствии резервной копии сертификата, которым эта резервная копия была защищена, делает невозможным восстановление.
- При использовании Availability Group необходимо иметь копии сертификата и Database Master Key на всех репликах.
- Если вы используете TDE выберите различные сертификаты или асимметричные ключи для шифрования базы и шифрования резервной копии.

Backup
Encryption –
Пример
синтаксиса

```
BACKUP DATABASE [AdventureWorks2012]
TO DISK = N'C:\DATA\Backup\AW2012.bak'
WITH COMPRESSION,
      ENCRYPTION
      (ALGORITHM = AES_128,
       SERVER CERTIFICATE = [MyServerCert])
```

Сжатие

Алгоритм шифрования

Сертификат

- Создание и резервное копирование master database key.
- Создание и резервное копирование сертификата.
- Создание зашифрованной резервной копии.
- Восстановление базы из резервной копии.

Что нового в
Database
Engine - New
Design for
Cardinality
Estimation

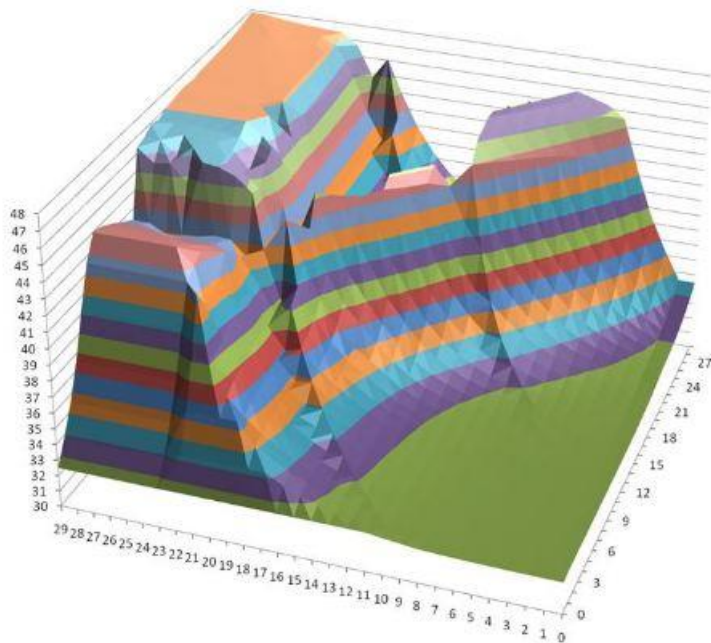
- Основная задача Cardinality Estimator (CE) – предсказать количество строк возвращаемых запросом.
- Основные проблемы с предыдущим методом:
 - Эффективность плана измеряется в терминах стоимости, который пропорционален времени выполнения запроса;
 - Оптимизатор запросов по сути непредсказуем и очень чувствителен к расчету Cardinality
 - Стоимость плана очень сильно зависит от количества строк возвращаемых каждым оператором плана выполнения;
 - Различия между стоимостью хорошего плана и случайно выбранным планом могут отличаться в десятки раз.

New Design for
Cardinality
Estimation –
Цели
улучшения

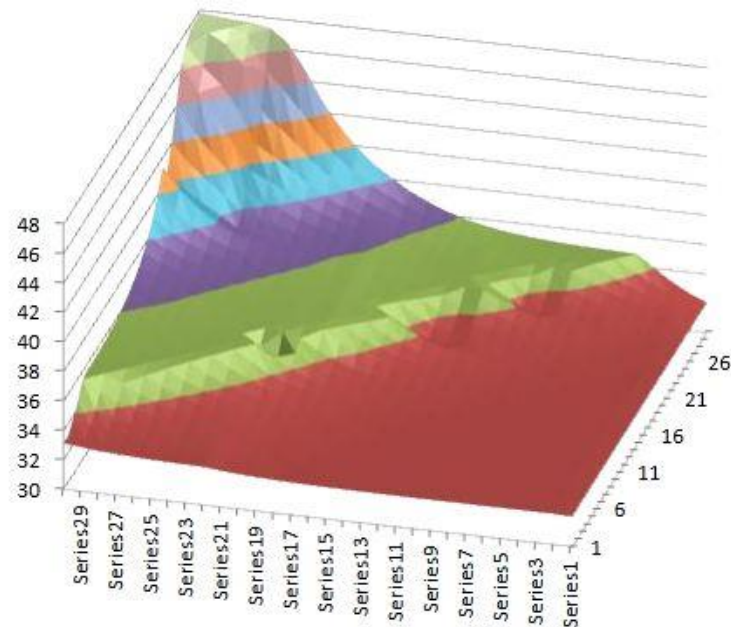
- Значительно расширить качество расчетов для широкого круга запросов и нагрузок (OLTP, DW)
- Увеличить производительность
- Сделать расчет более гладким и предсказуемым
- Версионность CE компонент
 - Возможно снижение производительности предыдущих запросов из-за изменения алгоритма;
 - Пользователи могут выбрать старый метод расчета используя флаги трассировки.

SQL Server 2008 R2

New Design for
Cardinality
Estimation –
Поверхность
расчета для
двух
параметров



Prototype with new CE



New Design
for Cardinality
Estimation –
Статистика –
основа
расчета (1)

```

Updated                Rows  Rows Sampled Steps Density      Average key length String Index
-----
Dec  2 2003 10:36AM 10000 10000          196 1.1331461E-3 12.0              NO

All density  Average Length Columns
-----
1.0638298E-3 4.0              a
1.0069479E-4 8.0              a, b
9.9999997E-5 12.0             a, b, c

(3 row(s) affected)

HI_KEY RANGE_ROWS EQ_ROWS DISTINCT_RANGE_ROWS AVG_RANGE_ROWS
-----
7      0.0        1.0      0              0.0
29     6.0        2.0      6              1.0
58     12.0       3.0      8              1.3333334
74     13.0       2.0     10              1.3
81     8.0        5.0      4              1.6
85     4.0        3.0      2              1.3333334
98     15.0       3.0     10              1.5
...     ...      ...      ...              ...
998    4273.0     22.0    242             17.657024
999     0.0       22.0      0              0.0

```

```

create table t (a int, b int, c int)
create statistics stat1 on t(a, b, c)
dbcc show_statistics(t, stat1)

```

New Design
for Cardinality
Estimation –
Статистика –
основа
расчета (2)

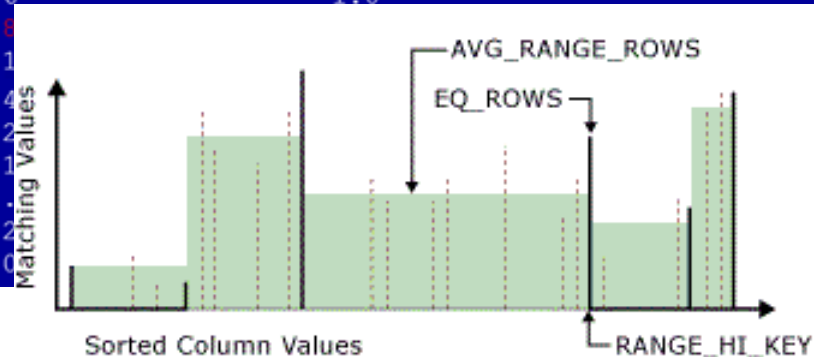
All density	Average Length	Columns
1.0638298E-3	4.0	a
1.0069479E-4	8.0	a, b
9.9999997E-5	12.0	a, b, c

Плотность (Density)

- $\text{Density} = 1 / (\text{Distinct_Value_Counts})$
- $\text{Average Frequency} = \text{Row Counts} * \text{Density}$

New Design
for Cardinality
Estimation –
Статистика –
основа
расчета (3)

HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
7	0.0	1.0	0	0.0
29	6.0	2.0	6	1.0
58	12.0	3.0	8	
74	13.0	2.0	1	
81	8.0	5.0	4	
85	4.0	3.0	2	
98	15.0	3.0	1	
...
998	4273.0	22.0	2	
999	0.0	22.0	2	



Histogram

- RANGE_HI_KEY
- EQ_ROWS
- RANGE_ROWS
- DISTINCT_RANGE_ROWS

New Design for
Cardinality
Estimation –
Основы
(Предположения)

Предположение об однородности набора (Uniformity Assumption):

- Уникальные значения существуют, равномерно распределены по набору и имеют одинаковую частоту встречаемости.

Предположение о наличии запрашиваемых данных (Containment Assumption):

- Пользователь запрашивает данные, которые есть в наборе.
- При объединении таблиц мы предполагаем, что значения, на основе которых производится объединение присутствуют в обоих объединяемых наборах.

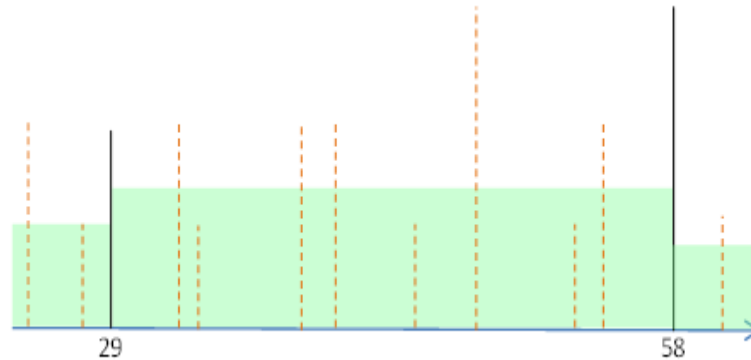
Предположение независимости (Independence Assumption):

- Распределения данных в различных столбцах независимы друг от друга.

New Design for Cardinality Estimation – Пример (1)

Select * from T where T.c1 = 50

- Значение 50 существует (containment)
- Частота значения 50 есть средняя частота (uniformity)

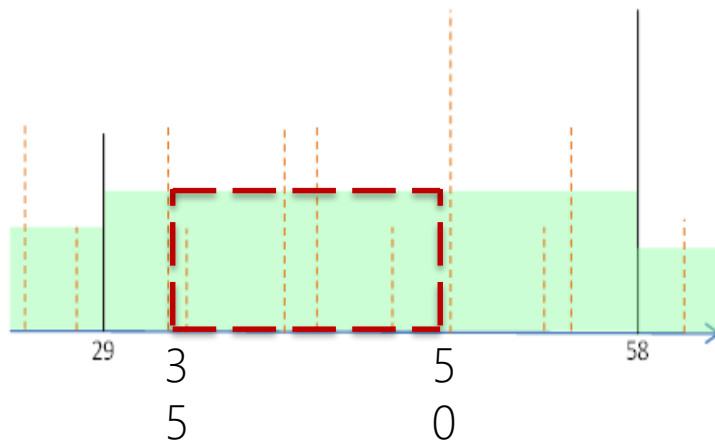


New Design for Cardinality Estimation – Пример (2)

Расчет по диапазону

Select * from T where T.c1 > 35 and T.c1 < 50

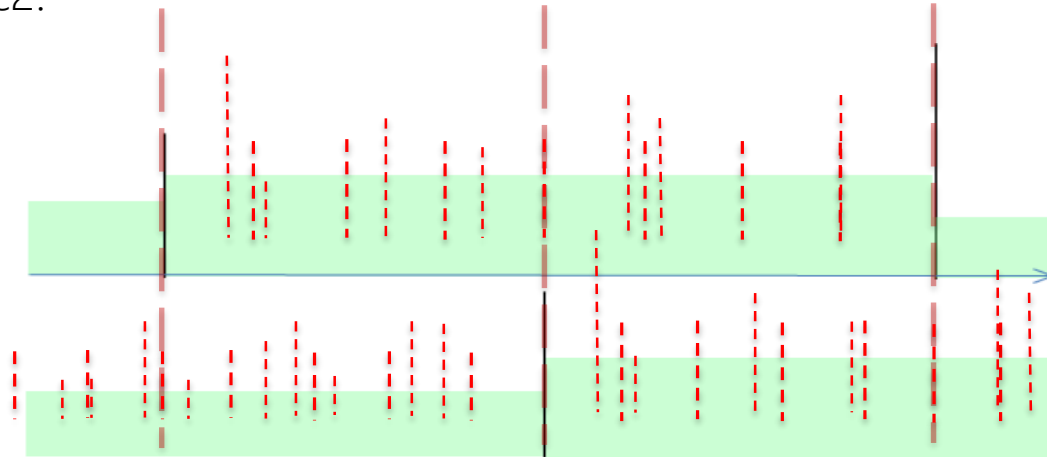
- Используется линейная интерполяция, что бы рассчитать количество возвращаемых строк и количество уникальных значений в диапазоне (35, 50) (Uniformity).



New Design
for Cardinality
Estimation –
Пример (3)

Select * from T1 join T2 on T1.c1 = T2.c2

- Расчет производится путем объединения гистограмм T1.c1 и T2.c2:



New Design
for Cardinality
Estimation –
Решение
проблем

- Флаги трассировки
 - 2312: Использовать новый механизм СЕ
 - 9481: Использовать старый механизм СЕ

New Design for
Cardinality
Estimation –
Демонстрация –
New Design for
Cardinality
Estimation.sql

Использование нового механизма СЕ

Использование старого механизма СЕ

Возможность подтверждения транзакции (commit) без ее записи в журнал транзакции.

Full transaction durability

- Транзакция записывается в журнал транзакций на диск перед возвращением управления клиенту.
- В такой системе никогда не возможна потеря данных.
- Возможна потеря производительности из-за задержки дисковых операций записи в файл журнала транзакций.

Delayed transaction durability

- Транзакция может не быть записана в журнал транзакций на диск, а управление клиенту будет возвращено.
- Производится асинхронная запись в журнал транзакций.
- Дисковые задержки никак не влияют на скорость выполнения операций модификации данных.

Delayed Durability (1)

Транзакция с Delayed durability (с отложенной фиксацией на диске) становится зафиксированной когда произойдет одно из этих событий:

- Транзакция, работающая в режиме нормальной фиксации (Fully Durable), будет зафиксирована на диске, что приведет к фиксации всех транзакций с отложенной фиксацией.
- Пользователь успешно вызовет хранимую процедуру `sp_flush_log`.
- Буфер памяти, где хранятся транзакции перед записью на диск (Log Buffer), будет переполнен, что приведет к его сбросу в журнал транзакций.

Delayed Durability (2)

Режим отложенной фиксации транзакций может использоваться для работы со:

- Стандартами базами данных (дисковыми)
- In-Memory базами данных.

Delayed Durability – Последствия

- Crash recovery (Некоторые незафиксированные изменения в ходе Crash Recovery могут быть потеряны)
- Cross-database and DTC (Гарантировано Durable)
- Always On Availability Groups and Mirroring (Может не быть Durable)
- Failover clustering (Некоторые незафиксированные изменения в ходе failover могут быть потеряны)
- Transaction Replication (Репликации подвергаются только Durable транзакции)
- Log shipping (Log shipping-гу подвергаются только Durable транзакции)
- Log Backup (Log backup-у подвергаются только Durable транзакции)

Delayed Durability – Синтаксис

ALTER DATABASE dbname SET DELAYED_DURABILITY = DISABLED |
ALLOWED | FORCED;

Разрешить на базе

Использовать для транзакции

COMMIT TRANSACTION WITH (DELAYED_DURABILITY = ON);

BEGIN ATOMIC WITH (DELAYED_DURABILITY = ON, ...)

Использовать для Native SP

Delayed
Durability –
Демонстрация –
Delayed
Durability.sql

Использование отложенной фиксации транзакций для при работе со стандартами таблицами (дисковыми).

Что нового в
Database
Engine -
AlwaysOn
Enhancements

- Добавлена возможность размещения реплик в Azure.
- Увеличено количество реплик с 4 до 8.
- Вторичная реплика остается доступна пользователям при потере кластером кворума и отсоединении от первичной реплики.
- SQL Server может использовать Cluster Shared Volumes (CSVs) как пространство для размещения файлов баз данных.
- Добавлены новые системные функции, [sys.fn_hadr_is_primary_replica](#) и [sys.dm_io_cluster_valid_path_names](#).
- Добавлена новая информация в [sys.dm_hadr_cluster](#), [sys.dm_hadr_cluster_members](#), [sys.dm_hadr_cluster_networks](#).

Что нового в Database Engine - Columnstore Indexes

- Новый формат хранения и алгоритм обработки данных.
- Данные хранятся не по строкам, а по столбцам, в специальных сегментах.
- Позволяет увеличить скорость обработки запросов более чем в 10...40 раз.
- Позволяет сжимать данные более чем 7 раз.
- В SQL 2012 были некластерные Columnstore индексы.
- В SQL 2014 появились кластерные Columnstore индексы.
- Из-за особенностей своей структуры кластерные Columnstore индексы преимущественно будут использоваться в системах анализа данных (Data Warehouse).

Columnstore Indexes - преимущества

- Данные в столбцах хорошо коррелируются – отсюда высокая степень сжатия
- Высокая степень сжатия экономит память, что позволяет всей таблице разместиться в RAM, что, в свою очередь, увеличивает скорость обработки запросов.
- Новый механизм работы с такими данными позволяет выполнять batch-обработку данных, что снижает нагрузку на процессор и увеличивает скорость обработки запроса в десятки раз.

Columnstore Indexes – Кластерный (SQL 2014)

- Доступен в Enterprise, Developer, Evaluation.
- Может обновляться.
- Первичный метод хранения данных
- Все столбцы включены в индекс.
- Не может быть комбинирован с другими индексами.
- Может быть сконфигурирован для использования columnstore или columnstore archival методов сжатия данных.
- Не сохраняет данные в сортированном виде.

Columnstore Indexes – Некластерный (SQL 2012)

- Может быть построен на “куче” или кластерном индексе только для части столбцов таблицы.
- Требуется дополнительное пространство для хранения.
- Для обновления требуется перестройка или переключение разделов. Не обновляется DML операторами (insert, update, delete)
- Может использоваться совместно с другими индексами на одной и той же таблице.
- Может быть сконфигурирован для использования с Columnstore или Columnstore archival сжатием.
- Данные, при создании индекса не сортируются, однако могут быть отсортированы для улучшения сжатия.

Columnstore Indexes – Термины

Columnstore index

- Технология для хранения, выборки и сопровождения данных используя по-столбцовый формат, называемый Columnstore. SQL Server поддерживает как кластерные, так и некластерные Columnstore индексы. Оба вида индексов оптимизированы для обработке данных в памяти.

Columnstore

- Данные, которые логически организованы как таблица, состоящая из строк и столбцов, но физически хранимых как страницы (сегменты) состоящие из столбцов.

Rowstore

- Данные логически организованы как строки, состоящие из столбцов, собранные в страницы построчно. Это стандартный способ хранения данных, используемый в SQL Server.

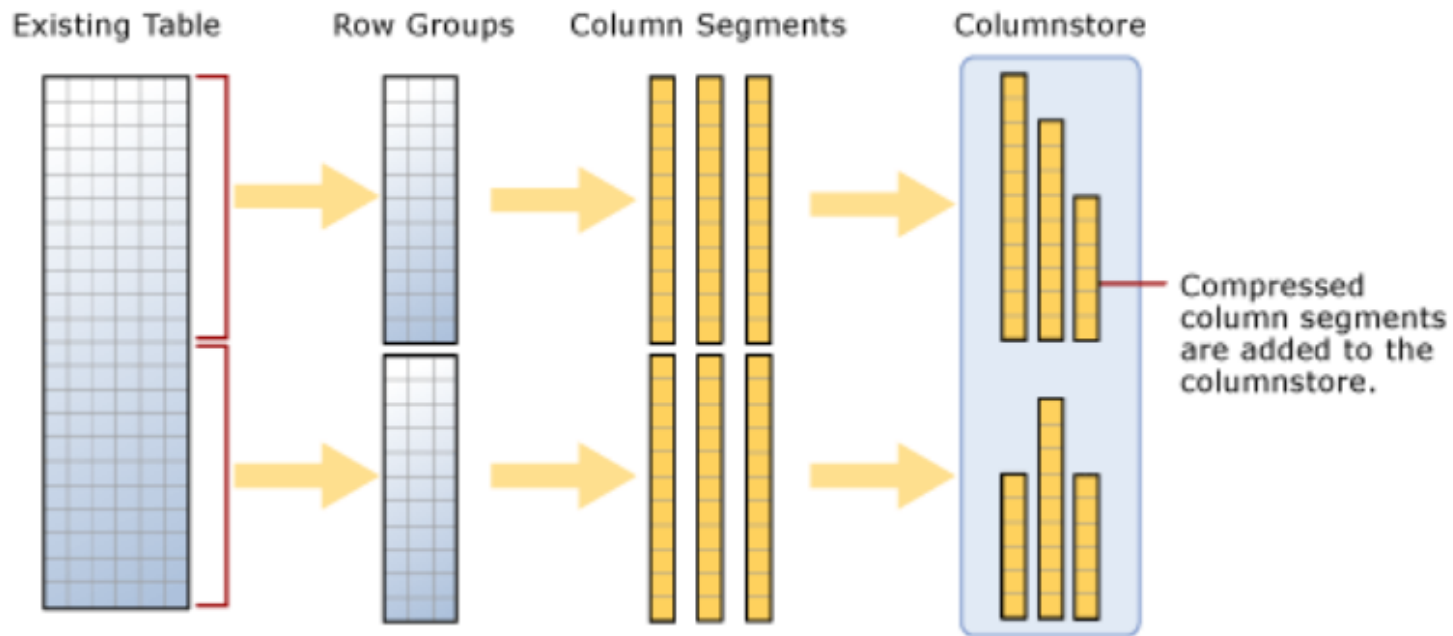
Rowgroups

- Группы строк, которые сжимаются и обрабатываются вместе. Обычно это 1,048,576 строк.

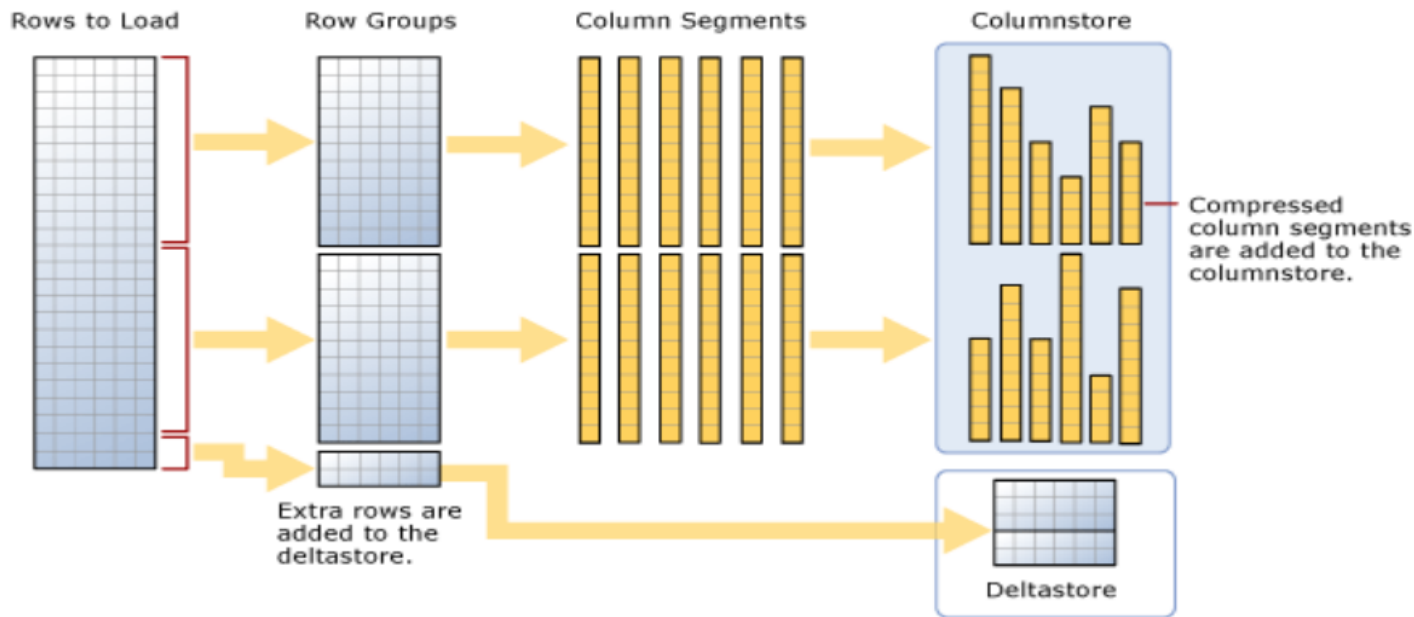
Column segments

- Сегменты памяти, содержащие только столбцы. Каждая группа строк содержит по одному сегменту для каждого столбца в таблице.
- Каждый сегмент столбцов сжимается вместе и сохраняется на физическом носителе.

Columnstore Indexes – Некластерные



Columnstore Indexes – Кластерные



Deltastore

Используется кластерным columnstore индексом только как временное хранилище вставленных строк и хранилище IDs для удаленных строк. Когда размер вставки превышает некоторый пороговый уровень (102 400 строк), данные перемещаются напрямую в Columnstore хранилище, при меньшем количестве – в Deltastore. Обработка идет блоками (Row Group). При выполнении запроса часть данных выбирается из Columnstore, а часть из Deltastore.

Columnstore Indexes – Особенности создание индекса

- Создание этого класса индекса - это параллельная операция, требующая большого объема памяти. Если памяти не хватает, то время построения индекса возрастает в разы.
- Объем памяти, необходимый для построения индекса, зависит от количества столбцов, количества строчных блоков, уровня параллелизма.
- Если сервер не имеет достаточного объема памяти для построения индекса в параллель, то сервер будет автоматически уменьшать уровень параллелизма, для того, что бы получить возможность построить индекс.

Что нового в
Database
Engine -
Columnstore
compression

Возможны два варианта сжатия:

- Columnstore compression.
- Columnstore archival compression.

Columnstore compression

Возможны новых два варианта сжатия:

- Columnstore compression.
- Columnstore archival compression.

Плюс ранее существовавшие:

- Row compression
- Page compression

Columnstore compression

- Columnstore данные всегда используют Columnstore compression (алгоритм VertyPaq).
- Данные могут быть дополнительно досжаты до Columnstore Compression Archival.
- Сжатие до уровня Columnstore Compression Archival потребует больше ресурсов при обращении к ним и поэтому запросы к ним могут выполняться медленнее, чем при уровне Columnstore compression.

Columnstore
compression –
Демонстрация
– ColumnStore
Indexes.sql

Что нового в Database Engine - Managing the Lock Priority of Online Operations

- Данная опция позволяет решить проблему невозможности Online перестройки или переключения разделов индексов при наличии блокировок наложенных пользователями
- При установлении опции `WAIT_AT_LOW_PRIORITY` сессия, производящая Online перестройку или переключение раздела, будет ожидать завершения этой операции время указанное в опции `MAX_DURATION`.
- Если сессия не сможет дожидаться и по истечении указанного интервала времени пользовательская блокировка не будет снята, то:
 - Сессия сама себя снимет с выполнения и Online перестройка индексов или переключение разделов производиться не будут.
 - Сессии пользователей, блокирующие данную сессию, будут сняты с выполнения.

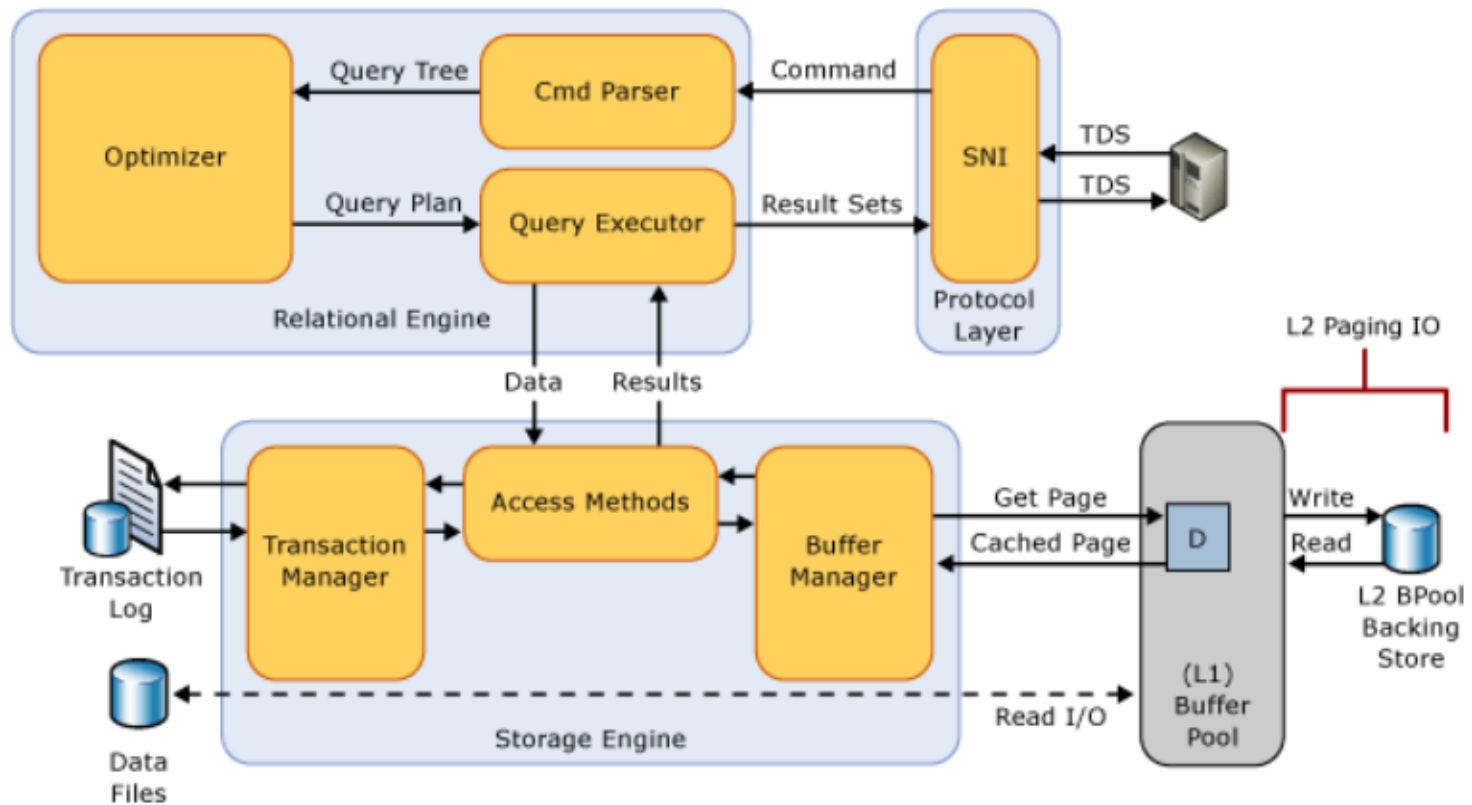
Что нового в Database Engine - Buffer Pool Extension

Buffer pool extension позволяет использовать для хранения части страниц данных, которые помещены в буферный пул, SSD диски. Перенаправляя часть страниц в это расширение буферного пула, увеличивается эффективность буферного пула и уменьшается время доступа к страницам памяти.

Используя данную опцию вы получаете новые возможности, а именно:

- Увеличить количество IOPS
- Уменьшить латентность I/O
- Увеличить транзактную производительность
- Увеличить производительность операций чтения данных
- Создать новую архитектуру кеширования данных для текущих и будущих приложений.

Buffer Pool Extension (1)



Buffer Pool Extension (2)

We recommend that you follow these best practices.

- Оптимальное значение для Buffer pool extension в 4...8 раз больше чем `max_server_memory`, хотя максимально может достигать 32.
- Тестируйте до применения в продуктивной среде. Поскольку отключение во время работы может сказаться на производительности выполнения запросов.
- Когда выключаете эту возможность память задействованная для ее реализации не освобождается до рестарта SQL Server.

Buffer Pool
Extension –
Демонстрация
– 10.sql

Проблемы стандартного обновления статистики:

- При добавлении новых данных в раздел (разделы) необходимо пересчитывать всю статистику, даже по разделам, которые не изменялись.
- Порог 20% рассчитывается не для модифицированного раздела, а для всей таблицы, что при больших размерах таблиц может никогда не наступить.
- Независимо от количества разделов собирается 200 дистрибутивных выборок для формирования итоговой статистики.

Incremental Statistics

- Первые две проблемы решены в SQL 2014 путем использования Инкрементальной статистики.
- При добавлении новых данных в раздел (разделы) нет необходимости пересчитывать всю статистику, можно пересчитать ее только по разделам, которые изменялись.
- Порог 20% теперь рассчитывается для каждого модифицированного раздела, а для всей таблицы

Incremental Statistics - ограничения

Создание инкрементальной статистики невозможно:

- на индексе, который не выровнен по разделу (not partition-aligned) с базовой таблицей.
- на AlwaysOn вторичных базах
- на read-only базах
- на фильтрованных индексах
- на view

Incremental
Statistics –
Демонстрация
– 11.sql

Что нового в
Database
Engine -
Resource
Governor
Enhancements
for Physical IO
Control

- Добавлена возможность управления операциями ввода/вывода для ресурсного пула.
- Возможно указать MIN_IOPS_PER_VOLUME – минимальное количество IOPS для данного пула на том.
- Возможно указать MAX_IOPS_PER_VOLUME – максимальное количество IOPS для данного пула на том.
- Значения по умолчанию – 0, что обозначает, что пул будет использовать максимально возможное количество операций ввода вывода.
- Если вы установили MIN_IOPS_PER_VOLUME в значение отличное от 0, то лучше установить MAX_IOPS_PER_VOLUME в максимальное значение (2,147,483,647).

Resource
Governor
Enhancements
for Physical IO
Control –
Демонстрация
– 12.sql

Что нового в Database Engine – System View Enhancements

`sys.dm_exec_query_profiles` – предназначена для мониторинга в реальном масштабе времени исполняющихся запросов.

`sys.column_store_row_groups` – предназначена для получения информации по сегментам Columnstore индексов

System View
Enhancements –
Демонстрация –
13.sql, 14.sql

Использование - sys.dm_exec_query_profiles

Что нового в
Database
Engine -
Database
Compatibility
Level

Уровень совместимости 90 более не может использоваться в SQL 2014.

Q & A



www.microsoft.com/microsoftservices

