Индексы
и
все, все, все

Часть 2

Давайте познакомимся

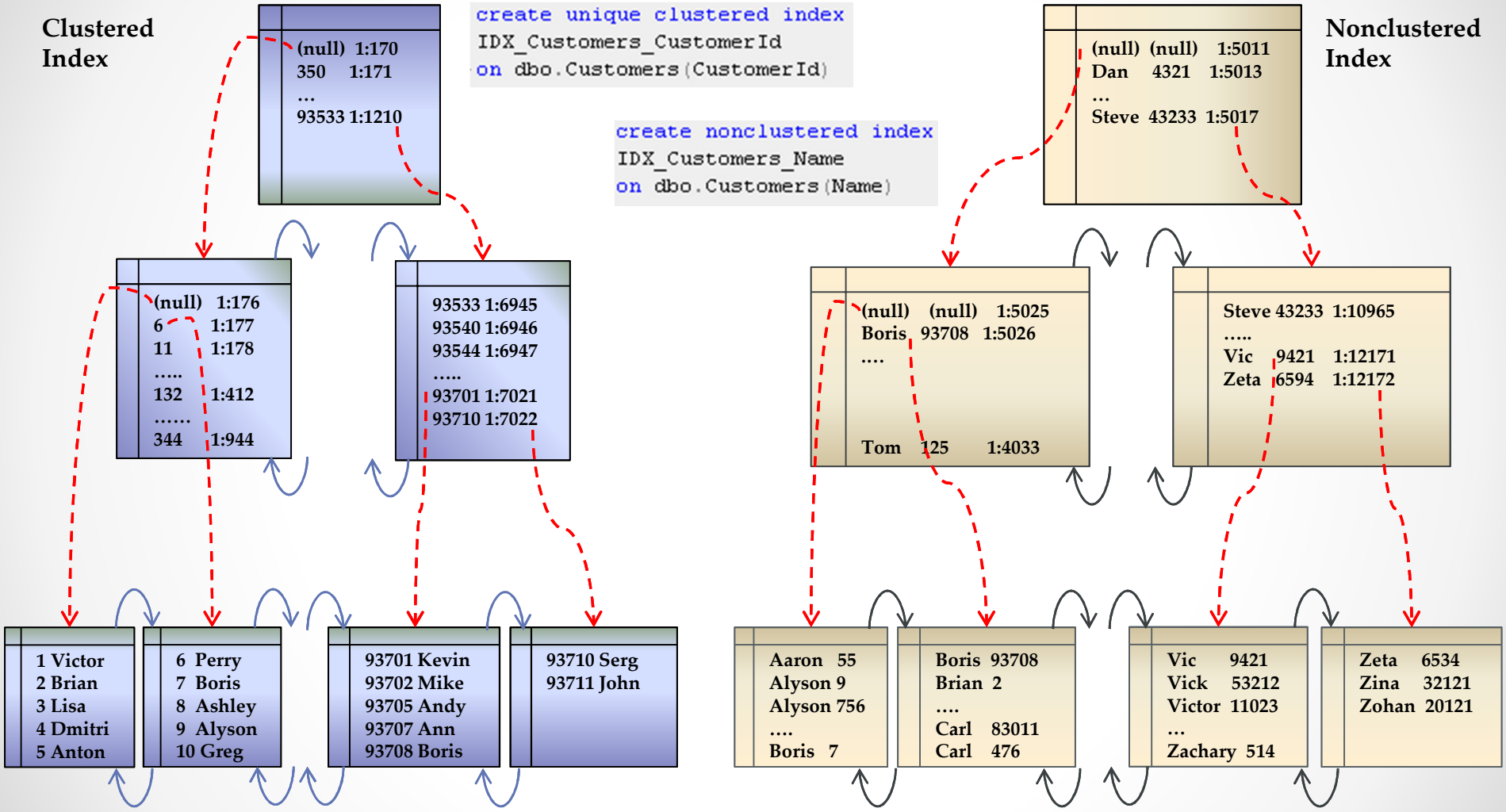
- 12+ лет опыта работы с Microsoft SQL Server
- Microsoft SQL Server MVP
- Microsoft Certified Master (SQL Server 2008)
- MCPD
  - Enterprise Application Developer

- Blog: http://aboutsqlserver.com
  - Презентация будет доступна для скачивания
- Email: dmitri@aboutsqlserver.com

Содержание

- Краткое содержание предыдущей серии
- Фрагментация и поддержка индексов
- Стратегии индексации
- Стратегии оптимизации

Структура индексов

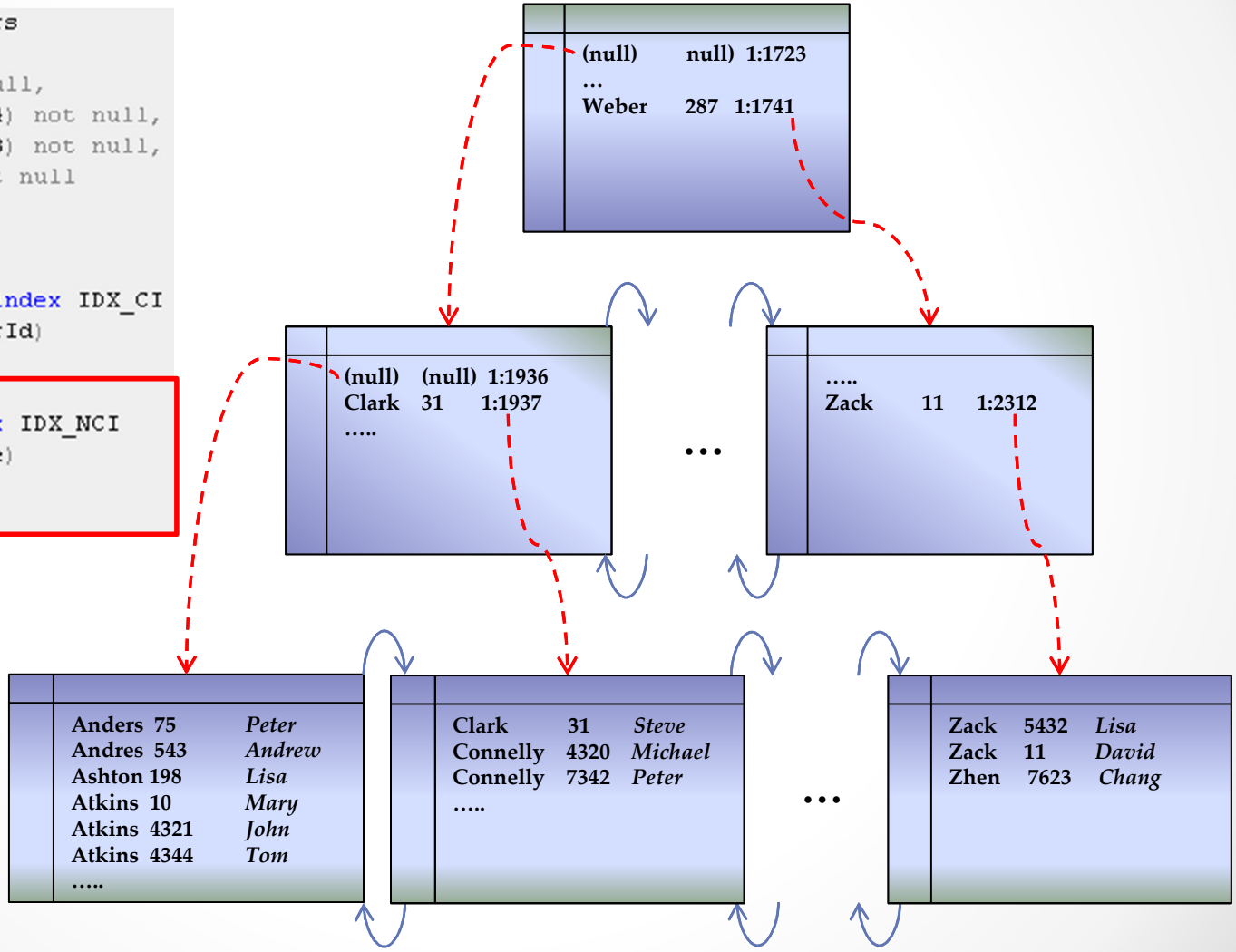


Индексы с включенными столбцами

```
create table dbo.Customers
(
    CustomerId int not null,
    FirstName nvarchar(64) not null,
    LastName nvarchar(128) not null,
    Phone varchar(32) not null
)
go

create unique clustered index IDX_CI
on dbo.Customers(CustomerId)
go

create nonclustered index IDX_NCI
on dbo.Customers(LastName)
include(FirstName)
go
```



Индексы с включенными столбцами

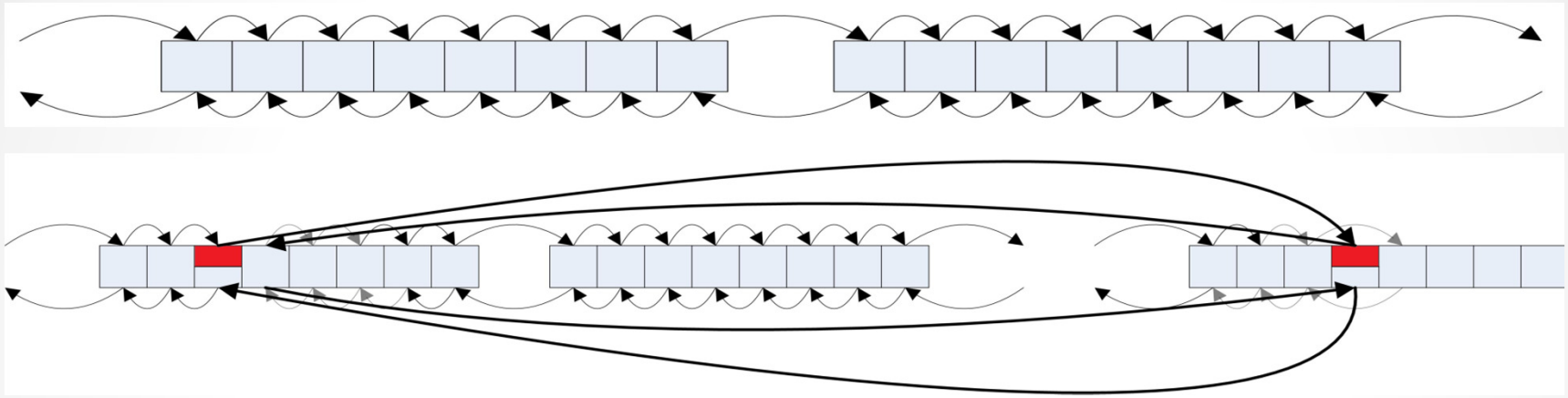
Демонстрация

Ключевые моменты

- Key lookup – очень дорогая операция
  - SQL Server не использует некластерные индексы в случае, если предполагает большое число найденных записей (key lookup операций)
- SQL Server использует статистику для оценки количества найденных записей
  - Статистика (гистограмма) хранит до 200х значений первого (левого) поля ключа
  - Статистика автоматически обновляется при изменении ~20% записей (значений ключа) таблицы
- Индексы с включенными столбцами помогают при оптимизации
  - Нет необходимости в операции key lookup

Фрагментация

Разрыв страниц и фрагментация



- SQL Server переносит ~50% данных на другую страницу
- FILLFACTOR позволяет зарезервировать место на странице, но:
  - Работает только на этапе: CREATE INDEX/ALTER INDEX REBUILD
  - Уменьшает разрыв страниц и фрагментацию, но увеличивает размер индекса

Паттерны, увеличивающие фрагментацию

Демонстрация

Паттерны, увеличивающие фрагментацию

- Insert/Update – увеличение размера записи
- Read Committed Snapshot / Snapshot
- **Триггеры**
- Индексы на “случайных” значениях
  - Uniqueidentifier
  - Hashbytes

Влияние фрагментации на систему

- Фрагментация наиболее опасна при сканировании данных (index SCAN)
  - Data Warehouse / Reporting / Decision Support Systems
  - Неоптимизированные запросы
- Фрагментация менее критична для операций поиска (index SEEK)
  - OLTP
- Фрагментация ухудшает производительность пакетных операций
- Большой процент свободного места на страницах увеличивает размер индекса и использует большее количество памяти в Buffer Pool.

Поддержка индексов

- sys.dm_db_index_physical_stats
  - **avg_fragmentation_in_percent**
  - **avg_page_space_used_in_percent**
- ALTER INDEX..REBUILD
  - Пересоздание индекса (и статистики)
  - (Опционально) Online операция в Enterprise Edition
  - Рекомендовано (*) при фрагментации > 30%
  - Генерирует большое количество записей в журнале транзакций
- ALTER INDEX..REORGANIZE
  - Логическая дефрагментация
  - Online операция
  - Рекомендовано (*) при фрагментации от 5% до 30%

(*) Books online – средняя температура по больнице

Дизайн стратегии поддержки индексов

Следует принять во внимание

- Тип системы (OLTP, DW, Смешанный)
- Требования к доступности системы (24x7x365?)
- Редакция SQL Server – можно ли использовать online index rebuild
- Есть ли другие БД на сервере
- Стратегии высокой доступности
  - Нагрузка на журнал транзакций

Ключевые моменты

- Избегайте индексы на “случайных значениях” ключа
- Избегайте паттерны, увеличивающие размер записи
  - Не используйте FILLFACTOR = 100
- Анализируйте систему в целом при разработке плана поддержки индексов

Стратегии Индексации

Кластерный Индекс

- Первичный ключ != Кластерный индекс
  - Primary key → элемент логического дизайна
  - Clustered Index → элемент физического дизайна
- Идеальный кластерный индекс
  - Уменьшает разрыв страниц и фрагментацию
  - Оптимизирует наиболее критичные запросы
- Физические характеристики идеального CI:
  - Статичность
    - При изменении ключа CI дата переносится в другое место в таблице + изменяются все записи в NCI ссылающиеся на измененные записи
  - Небольшой размер ключа
    - Значение ключа CI присутствует во всех NCI
  - Уникальный
    - SQL Server добавляет 4 байта *uniquifier int null* столбец для поддержки уникальности
      - Реально добавляется от 0 до 8 дополнительных байт

Неуникальные Кластерные Индексы

Демонстрация

Identity / Uniqueidentifiers / Sequences

- Статичны, уникальны, небольшие
- Потенциальные проблемы:
  - Горячие точки – сериализация при получении новых страниц / участков
  - Редко помогают при оптимизации запросов
- Область применения
  - Ссылочные таблицы (Articles, Customers, etc.)
  - В некоторых случаях отсутствие других кандидатов для CI
- Identity – 4/8 байт. Uniqueidentifiers – 16 байт
  - NEWID() – приводят к фрагментации

Uniqueidentifiers

Демонстрация

Логическое секционирование

```
create table dbo.Positions
(
    CompanyId int not null,
    UTCTimeTag datetime2(0) not null,
    RecId bigint not null,
    DeviceId int not null,
    Latitude decimal(9,6) not null,
    Longitude decimal(9,6) not null
)
go

create unique clustered index IDX_CI
on dbo.Positions
(CompanyId, UTCTimeTag, RecId)
go

create nonclustered index IDX_NCI
on dbo.Positions
(CompanyId, DeviceId, UTCTimeTag)
```

- Левый столбец идентифицирует секцию
  - Локализация I/O внутри секции
  - Уменьшение фрагментации при монотонном возрастании правых столбцов индекса
- Потенциальные проблемы
  - Статистика хранит гистограмму только по левому полю

Некластерные индексы – Уникальность

- Ограничение уникальности (unique constraint) != уникальному индексу
  - Unique Constraint → компонент логического дизайна
  - Unique Index → компонент физического дизайна
  - SQL Server использует уникальные индексы в обоих случаях
- Структура уникального индекса более оптимальна
  - Промежуточные и корневой уровни не содержат *row id*
- Уникальность помогает оптимизатору запросов

Некластерные индексы – Использование

- SQL Server *редко* использует >1 некластерного индекса на таблицу в одном запросе за исключением:
  - *Сканирование данных == неоптимизированный запрос*
  - *OR предикаты*
- Композитный индекс с включенными столбцами обычно более эффективен, нежели набор индексов состоящих из одного столбца
  - Необходимо принять во внимание дополнительную нагрузку при модификации данных и поддержку индексов

Сколько индексов на таблицу?

- OLTP системы
  - Множество небольших транзакций, выполняющихся одновременно
  - Данные постоянно обновляются
  - Простые и оптимизированные запросы
    - Минимально необходимый набор индексов
- Data Warehouse системы
  - Небольшое количество тяжелых запросов (обычно сканирование), работающих одновременно
  - Данные обновляются по расписанию
    - Число индексов зависит от запросов и стратегии обновления данных
      - Удаление и пересоздание индексов во время загрузки
      - Индексируемые представления
      - ColumnStore индексы

Ключи размером > 900 байт

- Индекс на хранимом калькулированном столбце - persisted calculated column - (CHECKSUM, HASHBYTES)
- Обязательно использовать оригинальный столбец в запросах

Ключи размером > 900 байт

Демонстрация

Ключевые моменты

- Создавайте кластерные индексы уникальными, небольшими и статичными
- Избегайте индексы на полях «со случайными значениями» (guid, hashbyte)
- Используйте индексы с включенными столбцами

Стратегии оптимизации

Стратегии оптимизации

Разработка систем с нуля

- Шаг 1 – создание минимально необходимого набора индексов:
  - Первичные ключи и кластерные индексы
  - Поддержка уникальности
  - Поддержка ссылочной целостности
  - Индексы для наиболее важных запросов (когда известны)

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - Удаление неиспользованных индексов
  - Консолидация
  - Анализ проблематичных индексов

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - **Удаление избыточных индексов**
  - Удаление неиспользованных индексов
  - Консолидация
  - Анализ проблематичных индексов

```
create nonclustered index IDX_1
on dbo.Customers(LastName, FirstName)
```

```
create nonclustered index IDX_2
on dbo.Customers(LastName)
```

- Нет правил без исключений
  - Иногда размер ключа имеет значение

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - **Удаление неиспользованных индексов**
  - Консолидация
  - Анализ проблематичных индексов

```
select *
from sys.dm_db_index_usage_stats us
where
    us.database_id = DB_ID() and
    us.object_id = OBJECT_ID(N'dbo.Orders')
order by
    us.index_id
```

index_id	user_seeks	user_scans	user_lookups	user_updates	last_user_seek	l
1	27341	2578	28123	9921	2012-03-23 15:27:10.270	:
2	5368	0	0	299	2012-03-23 15:26:55.697	I
3	302	0	0	46	2012-03-23 15:21:06.557	I
5	1677	0	0	49	2012-03-23 15:27:05.317	I
8	21065	0	0	49	2012-03-23 15:27:08.737	I
50	0	0	0	28	NULL	I
51	0	2382	0	60	NULL	:

- sys.dm_db_index_operational_stats – дополнительная информация
- **Данные очищаются при перезапуске SQL Server**

Неиспользуемые индексы

Демонстрация

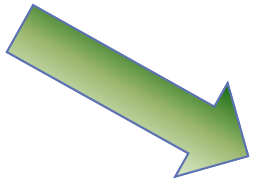
Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - Удаление неиспользованных индексов
  - **Консолидация**
  - Анализ проблематичных индексов

```
create nonclustered index IDX_1
on dbo.Customers(LastName, FirstName)
include(DateOfBirth)

create nonclustered index IDX_2
on dbo.Customers(LastName)
include(Phone)
```



```
create nonclustered index IDX_3
on dbo.Customers(LastName, FirstName)
include(Phone, DateOfBirth)
```

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - Удаление неиспользованных индексов
  - **Консолидация**
  - Анализ проблематичных индексов

```
create nonclustered index IDX_4
on dbo.Orders(OrderDate, Total)

create nonclustered index IDX_5
on dbo.Orders(OrderDate, WarehouseId)
```

```
create nonclustered index IDX_6
on dbo.Orders(OrderDate, Total)
include(WarehouseId)
```

```
create nonclustered index IDX_7
on dbo.Orders(OrderDate, WarehouseId)
include(Total)
```

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - Удаление неиспользованных индексов
  - Консолидация
  - **Анализ проблематичных индексов**

```
select *
from sys.dm_db_index_physical_stats(DB_ID(),Object_Id(N'dbo.Orders'),null, null, 'DETAILED')
```

index_id	avg_fragmentation_in_percent	fragment_count	avg_fragment_size_in_pages	page_count	avg_page_space_used_in_percent
3	98.5034013605442	732	1.00409836065574	735	64.8512972572276
51	3.53982300884956	19	23.7894736842105	452	97.8963800345935

Стратегии оптимизации

Оптимизация существующих систем

- Шаг 1
  - Удаление избыточных индексов
  - Удаление неиспользованных индексов
  - Консолидация
  - **Анализ проблематичных индексов**

```
select *
from sys.dm_db_index_operational_stats(DB_ID(),Object_Id(N'dbo.Users'),null, null)
```

index_id	leaf_insert_count	leaf_update_count	singleton_lookup_count	range_scan_count	row_lock_count	page_io_latch_wait_count	page_io_latch_wait_in_ms
1	18	27457	326612	0	88905	172	2782
2	307	0	0	58715693	1158	207	2075
3	18	0	304	0	36	17	673
5	23	0	0	1767	51	9	329
8	20	3	1471	257	148	11	516
50	0	0	0	0	0	0	0
51	125	0	0	0	357	4	62
100	307	0	0	42526	903	7	514

Стратегии оптимизации

- **Поиск проблематичного кода**
  - Data Management View
  - SQL Profiler
  - Performance Data Collectors / MDW
  - 3rd Party tools
  - И.т.д
- Анализ проблематичного кода
  - Can code be simplified and/or refactored?
  - Is Statistics up to date?
  - Does recompilation help (parameter sniffing)?
- Добавление необходимых индексов

Инструменты

- Missing Indexes DMVs
  - Рекомендует индексы, которые помогают текущему запросу и только текущему плану
  - Не принимает во внимание никаких дополнительных факторов
- Database Tuning Advisor
  - Качество зависит от трейса
- Могут помочь найти проблематичные места

Sys.dm_exec_query_stats

```
SELECT TOP 250
    SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
        ((
            CASE qs.statement_end_offset
                WHEN -1 THEN DATALENGTH(qt.TEXT)
                ELSE qs.statement_end_offset
            END - qs.statement_start_offset)/2)+1) as SQL,
    qs.execution_count as [Exec Cnt],
    (qs.total_logical_reads + qs.total_logical_writes) / qs.execution_count as [Avg IO],
    qp.query_plan,
    qs.total_logical_reads as [Total Reads],
    qs.total_logical_writes as [Total Writes],
    qs.total_worker_time as [Total CPU],
    qs.total_rows, qs.last_rows, qs.min_rows, qs.max_rows,
    qs.last_worker_time, qs.last_logical_reads, qs.last_logical_writes,
    qs.total_elapsed_time/1000 total_elapsed_time_in_ms,
    qs.last_elapsed_time/1000 last_elapsed_time_in_ms,
    qs.last_execution_time
FROM
    sys.dm_exec_query_stats qs with (nolock)
        CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
        OUTER APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY
    [Avg IO] desc
option (recompile)
```

!! Работает только с закэшированными запросами !!

Sys.dm_exec_query_stats

```
SELECT TOP 250
    SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
        ((
            CASE qs.statement_end_offset
                WHEN -1 THEN DATALENGTH(qt.TEXT)
```

	SQL	Exec ...	Avg IO	query_plan	Total Reads	Total Writes	Total CPU
1	select Subj, cast(R...	1	6816382	<ShowPlanXM...	6816296	86	24297389
2	select UID, DOCTY...	26455	4143503	<ShowPlanXM...	109616393555	0	154369131409
3	DELETE TOP (@d...	1	4096631	<ShowPlanXM...	4096468	163	26538518
4	insert into #tmpRep...	62	3690210	NULL	228750206	42859	3351099613
5	update #tmpReportI...	62	3139967	NULL	194677952	7	2406888686
6	insert into #tmpRep...	58	2516483	NULL	145905711	50341	1761652781
7	select D.*, O.CATE...	16	1848720	<ShowPlanXM...	29579527	0	64629691
8	update #tmpReport...	13	1520333	<ShowPlanXM...	19764334	5	194722131
9	select D.*, I.Catego...	36	1511917	<ShowPlanXM...	54429042	0	114735561
10	update #tmpReport...	26	1459946	<ShowPlanXM...	37958482	138	447010567
11	update #tmpReport...	12	1426777	<ShowPlanXM...	17121325	4	164099386
12	insert into #tmpRep...	53	1079374	NULL	57198359	8467	865721533

```
ORDER BY
    [Avg IO] desc
option (recompile)
```

!! Работает только с закэшированными запросами !!

Ключевые моменты

- Избавляйтесь от «плохих» индексов перед созданием «хороших» индексов
- Будьте осторожны с инструментарием
- Sys.dm_exec_query_stats – замечательный инструмент для анализа

Мы обсудили

- Фрагментация и поддержка индексов
- Стратегии индексации
- Стратегии оптимизации

Вопросы?

- Огромное спасибо за внимание!

- Email: dmitri@aboutsqlserver.com
- Blog: http://aboutsqlserver.com