

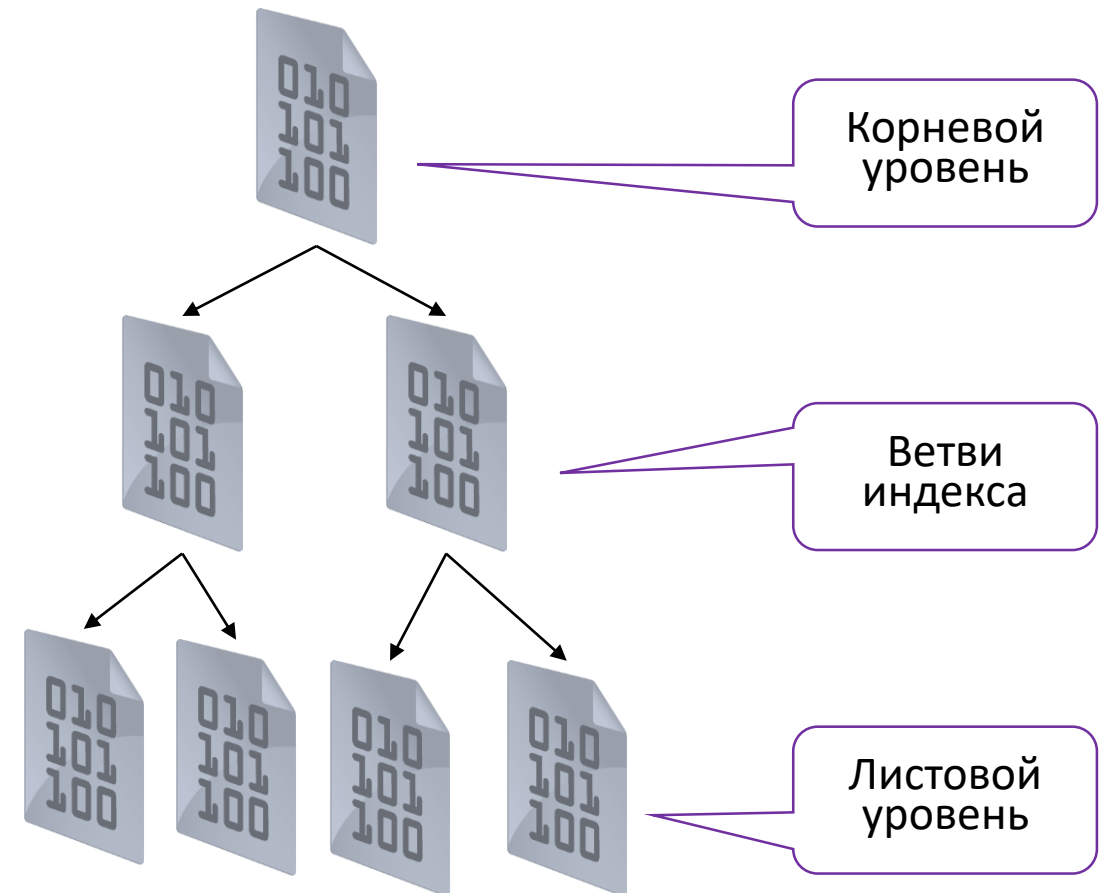
UNDERSTANDING INDEXES

О ЧЕМ ПОГОВОРИМ

- Что такое индексы
- Кластерные и некластерные индексы
- Покрывающие индексы и индексы типа INCLUDE
- Сбор информации из динамических представлений (DMVs)
- Индексированные (материализованные) представления
- Фильтрованные индексы
- Гипотетические индексы SQL Server
- Статистика
- Поддержка индексов

Что такое индексы

- SQL Server обращается к данным, сканируя таблицу или индекс
 - Сканируя таблицу, SQL Server читает все страницы
 - Сканируя индекс, SQL Server использует страницы индекса для поиска нужных строк
- Индексы могут быть *кластерными* и *некластерными*
- Индексы можно создать отдельной командой или определив PRIMARY KEY



Свойства индекса

- > Количество операций I/O (ввода/вывода) зависит от высоты индекса
- > Корневой узел и узлы — ветви индекса сжимаются и поэтому содержат ровно столько начальных байтов значения ключа, сколько нужно для того, чтобы отличить его от других значений. Листовой уровень содержит полное значение ключа
- > Значения в индексе упорядочиваются по ключевому значению
- > Индекс можно использовать для поиска как точного соответствия, так и диапазона значений
- > Составной ключ не будет применяться, если лидирующая часть составного ключа не совпадает с перечнем полей в запросе (например, в предложении WHERE)
- > СУБД обычно сама принимает решение, использовать индекс или нет. Значения колонок NULL не индексируются.

Кластерные индексы

- Строки хранятся в логическом порядке
- Только 1 кластерный индекс на таблицу
- Таблицу без кластерного индекса называют кучей (a heap)
- Все данные в индексе хранятся на листовом уровне
- Аналог книги вместе с оглавлением
- С заданным физическим порядком содержимого

Кластерные индексы

> Вставка (INSERT)

- Новая строка должна быть помещена в правильную логическую позицию
- Может приводить к расщеплению страниц таблицы

> Выборка (SELECT)

- Запросы с выборкой по кластерному индексу работают быстрее
- Запросы не требуют сортировки по полю (полям) кластерного индекса

> Удаление (DELETE)

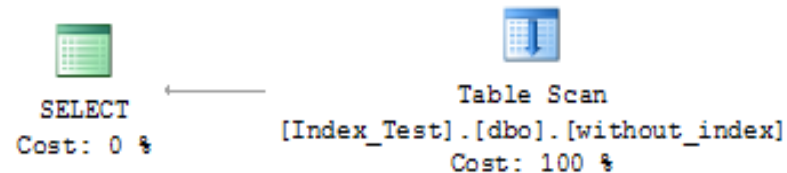
- Освобождает пространство, помечая данные как неиспользуемые

> Обновление (UPDATE)

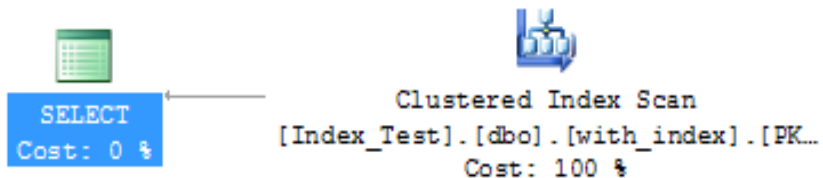
- Новая строка может остаться на той же позиции, если она помещается на страницу и значение кластерного ключа не изменилось
- Если строка больше не помещается на странице, она будет расщеплена
- Если кластерный ключ изменился, то строка будет удалена и помещена в нужную позицию согласно новому заданному порядку

Работая с кластерными индексами

Query 1: Query cost (relative to the batch): 50%
SELECT * FROM dbo.without_index



Query 2: Query cost (relative to the batch): 50%
SELECT * FROM dbo.with_index



SELECT

```
    i.name AS index_name
    ,i.type_desc --тип индекса (кластерный,
некластерный, куча)
    ,is_unique
    ,ds.type_desc AS filegroup_or_partition_scheme
    ,ds.name AS filegroup_or_partition_scheme_name
    ,ignore_dup_key --1 = IGNORE_DUP_KEY is ON
    ,is_primary_key --1 = Индекс является частью PK
    ,is_unique_constraint --1 = Индекс - часть Unique
    ,fill_factor
    ,is_padded --1 = PADINDEX is ON
    ,is_disabled --1 = Индекс отключен
    ,allow_row_locks
    ,allow_page_locks
FROM sys.indexes AS i
INNER JOIN sys.data_spaces AS ds ON i.data_space_id =
ds.data_space_id
WHERE is_hypothetical = 0
AND i.index_id <> 0
AND i.object_id = OBJECT_ID('Production.Product');
```

Некластерные индексы

- Таблица по структуре либо куча (heap), либо кластерный индекс
 - Куча (Heap) = Index ID 0
 - Кластерный индекс = Index ID 1
- Можно создать дополнительные индексы
 - Они называются некластерными (Index ID 2+)
 - Отдельный объект по отношению к таблице
 - Листовой уровень содержит указатель на место в таблице, где могут быть найдены данные

Некластерные индексы на основе кучи (Heap)

id	index_id >= 2	root_page
----	---------------	-----------



Корневой узел индекса

Страницы индекса

Листовой уровень
Содержит Row IDs



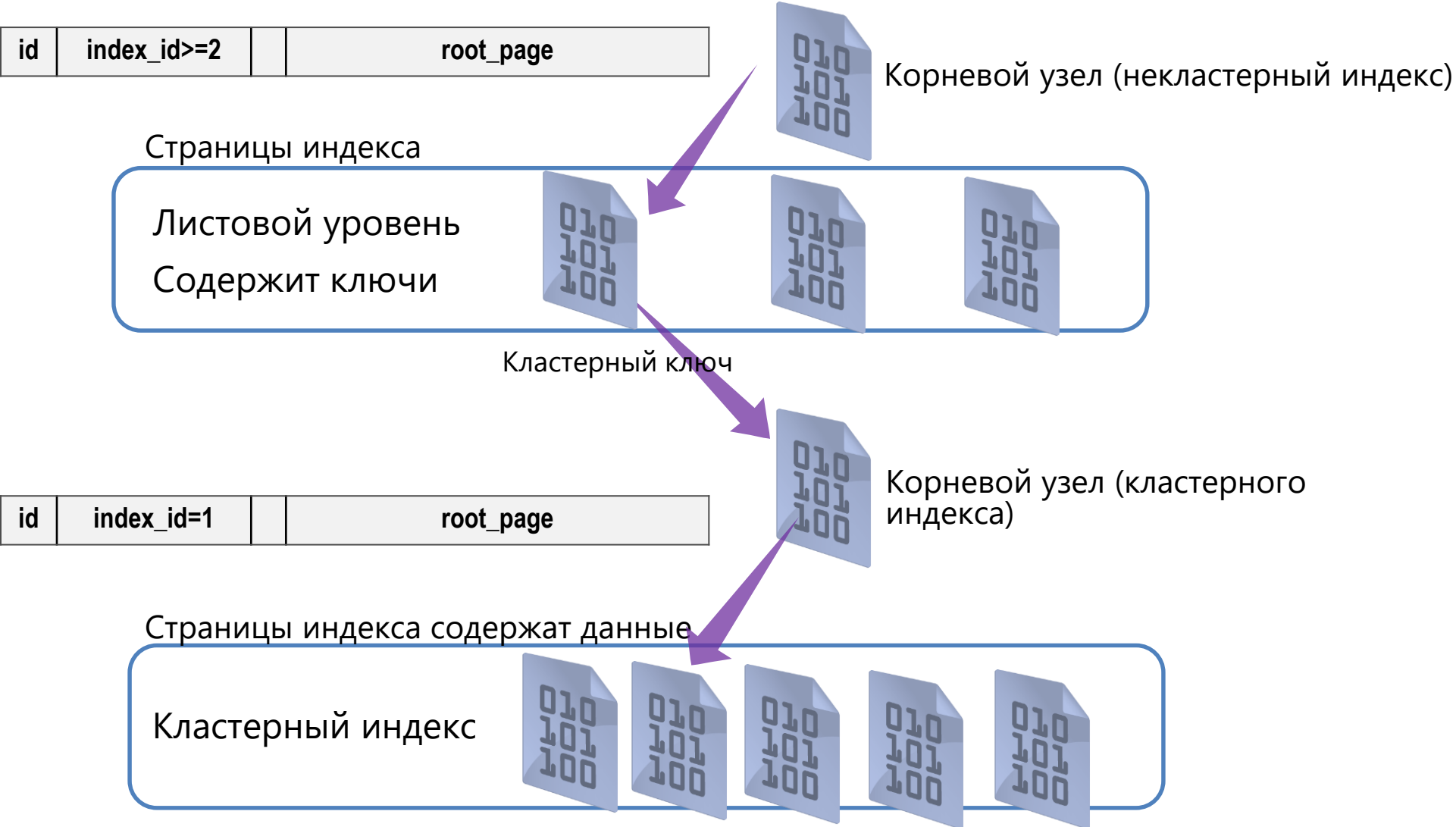
id	index_id = 0
----	--------------

Страницы с данными

Куча (Heap)

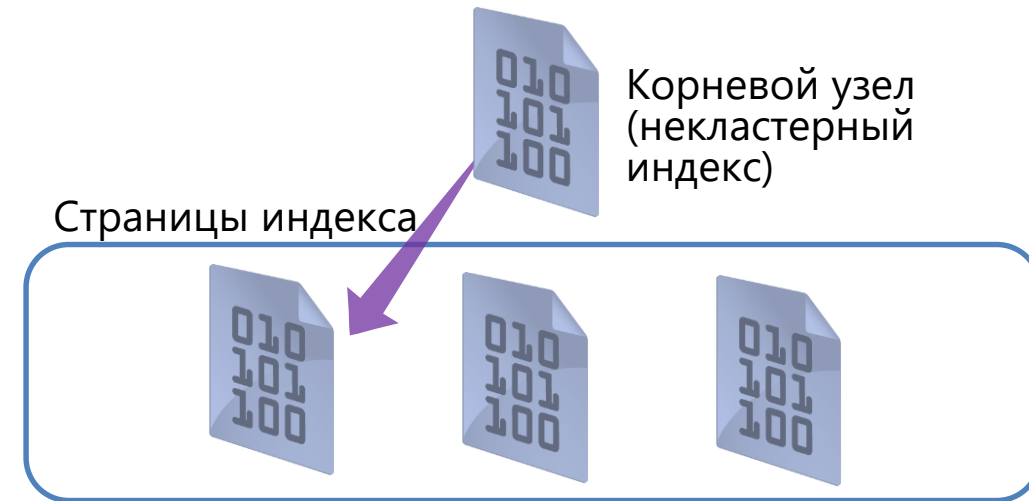


Некластерные индексы на основе Кластерного индекса



Покрывающие индексы и индексы типа INCLUDE

- > Покрывающий индекс – это индекс, который может предоставить все необходимые данные по запросу
 - улучшает производительность, так как исключает необходимость лукапить данные в таблице
 - до версии SQL Server 2005 можно было только создавать индексы, построенные по всем необходимым для запроса колонкам
 - сейчас доступны INCLUDE индексы, используются для вставки данных необходимых колонок на листовый уровень некластерного индекса



Листовой уровень содержит все колонки, перечисленные в SELECT (не нужно лукапить страницы кучи (heap) или кластерного индекса)

Работа с кластерными индексами

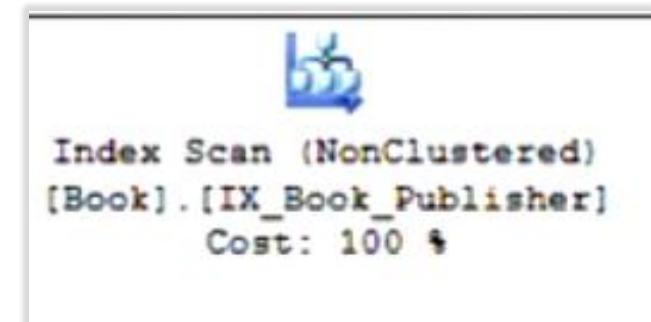
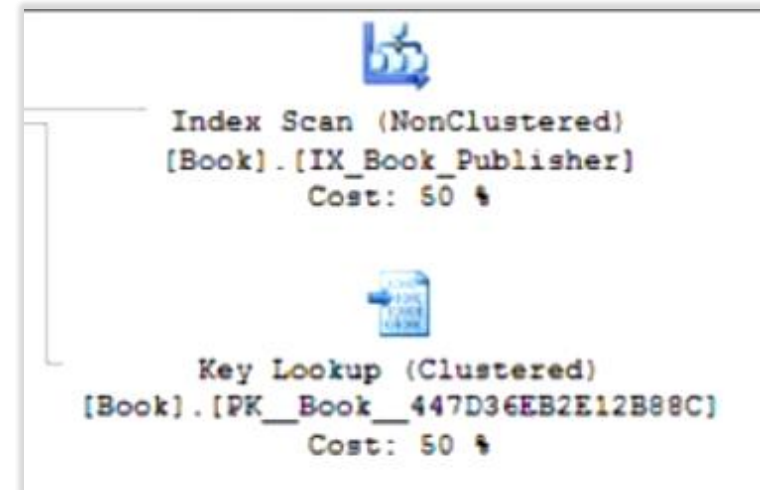
```
CREATE TABLE dbo.Book  
( ISBN nvarchar(20) PRIMARY KEY,  
  Title nvarchar(50) NOT NULL,  
  ReleaseDate date NOT NULL,  
  PublisherID int NOT NULL  
);  
GO
```

```
CREATE NONCLUSTERED INDEX IX_Book_Publisher  
  ON dbo.Book (PublisherID, ReleaseDate DESC);  
GO
```

```
CREATE NONCLUSTERED INDEX IX_Book_Publisher  
  ON dbo.Book (PublisherID, ReleaseDate DESC)  
  INCLUDE (Title)  
  WITH DROP_EXISTING;  
GO
```

 **accenture**

```
SELECT PublisherID, Title, ReleaseDate  
FROM dbo.Book  
WHERE ReleaseDate > DATEADD(year,-1,SYSDATETIME())  
ORDER BY PublisherID, ReleaseDate DESC;  
GO
```



Динамические представления (DMVs)

> `sys.dm_db_index_physical_stats`

- Статистика по размеру индекса и уровню фрагментации

> `sys.dm_db_index_operational_stats`

- Текущая статистика I/O по индексам и таблицам

> `sys.dm_index_usage_stats`

- Статистика по использованию индекса

Собрать информацию по индексам с помощью динамических представлений (DMVs)

-- Определить уровень фрагментации

```
SELECT * FROM
```

```
sys.dm_db_index_physical_stats(DB_ID(),OBJECT_ID('dbo.PhoneLog'),NULL,NULL,'DETAILED');
```

```
GO
```

-- Чтение и запись для поиска неиспользуемых индексов

```
SELECT convert(varchar(120),object_name(ios.object_id)) AS [Object Name],
```

```
       i.[name] AS [Index Name],
```

```
       SUM (ios.range_scan_count + ios.singleton_lookup_count) AS 'Reads',
```

```
       SUM (ios.leaf_insert_count + ios.leaf_update_count + ios.leaf_delete_count) AS
```

```
'Writes'
```

```
FROM   sys.dm_db_index_operational_stats (db_id(),NULL,NULL,NULL ) ios
```

```
INNER JOIN sys.indexes AS i
```

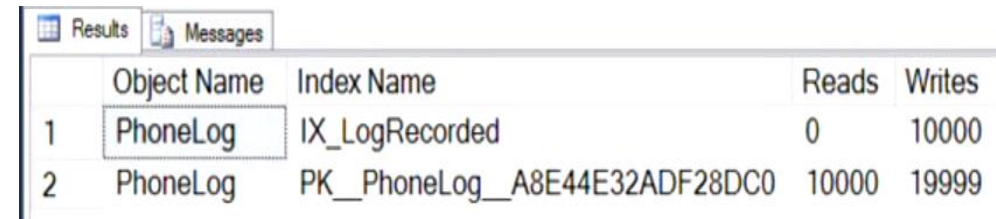
```
       ON i.object_id = ios.object_id
```

```
       AND i.index_id = ios.index_id
```

```
WHERE  OBJECTPROPERTY(ios.object_id,'IsUserTable') = 1
```

```
GROUP BY object_name(ios.object_id),i.name
```

```
ORDER BY Reads ASC, Writes DESC
```



The screenshot shows a SQL Server query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with four columns: 'Object Name', 'Index Name', 'Reads', and 'Writes'. There are two rows of data. The first row shows 'PhoneLog' as the object name and 'IX_LogRecorded' as the index name, with 0 reads and 10000 writes. The second row shows 'PhoneLog' as the object name and 'PK_PhoneLog_A8E44E32ADF28DC0' as the index name, with 10000 reads and 19999 writes.

	Object Name	Index Name	Reads	Writes
1	PhoneLog	IX_LogRecorded	0	10000
2	PhoneLog	PK_PhoneLog_A8E44E32ADF28DC0	10000	19999

Индексированные (материализованные) представления

- > в отличие от индекса на таблицу, все поля, включенные в представление, включены в данные индекса
- > Ограничения (следующие элементы T-SQL запрещены):
 - повторяющиеся колонки – например, SELECT Col1, Col2, Col1 AS Col (только если это часть различных выражений)
 - подзапросы.
 - ROWSET; UNION; TOP и ORDER BY; DISTINCT;
 - COUNT(*). Но можно COUNT_BIG(*)
 - Агрегатные функции: AVG, MAX, MIN, STDEV, STDEVP, VAR, VARP

Создать представление

```
CREATE VIEW ContractJobs WITH SCHEMABINDING  
AS  
SELECT .. FROM ..
```

Добавить уникальный класт. инд

```
CREATE UNIQUE CLUSTERED INDEX  
IX_ContractJobs_JobId ON ContractJobs (JobID)
```

Добавить доп. индексы

```
CREATE INDEX IX_ContractJobs_ContractNumber ON  
ContractJobs (ContractNumber)
```

Индексированные (материализованные) представления

> Ограничения (продолжение):

- включить опцию ANSI_NULLS в момент создания таблиц, ссылающихся на представление
- включить опцию ANSI_NULLS и QUOTED_IDENTIFIER до создания представления
- создать представление и любую пользовательскую функцию (UDF), используемую в представлении, с опцией SCHEMABINDING
- выборка полей только из базовых таблиц (не представлений) в той же БД с тем же владельцем
- ссылка на все таблицы и пользовательские функции только через составные имена (схема.таблица)
- поля в представлении должны быть перечислены в явном виде (без *)
- в запросе можно использовать только внутренние соединения (inner joins)
- все функции, используемые в представлении, должны быть детерминированными

Фильтрованные индексы

- > Фильтрованные индексы используют предложение WHERE для ограничения количества строк, входящих в индекс
- > Преимущества фильтрованных индексов
 - быстрая скорость ответа
 - низкие требования к дисковому пространству
 - более быстрое перестроение индекса

```
CREATE NONCLUSTERED INDEX
    NC_EMP_ADDRESS
ON HR.Address
(
    AddressLine1,
    AddressLine2
)
WHERE City='New York'
```

Гипотетические индексы в SQL Server

- > Когда хотим понять, насколько повысится производительность запроса после добавления нового индекса
- > DTA (Database Tuning Advisor) использует, чтобы рекомендовать пропущенные индексы
- > опция `STATISTICS_ONLY = -1` означает, что будет создан не сам индекс, а только статистика по нему. Этот индекс не будет использоваться оптимизатором до тех пор, пока вы не запустите запрос в режиме `AUTOPILOT` (`DBCC AUTOPILOT` и `AUTOPILOT MODE`) (см [полезные ссылки](#) для большей информации)

```
CREATE INDEX ixOrderDate  
ON  
Sales.SalesOrderHeader  
(OrderDate) WITH  
STATISTICS_ONLY = -1
```

Фрагментация индексов

- > Фрагментация происходит в тот момент, когда новые данные приводят к расщеплению страниц
 - Внутренняя фрагментация, когда страница не заполнена до конца
 - Внешняя фрагментация, когда страницы находятся не в логическом порядке
- > Как определить фрагментацию
 - свойства индекса в SQL Server Management Studio
 - `sys.dm_db_index_physical_stats`

FILLFACTOR и PAD_INDEX

```
ALTER TABLE Person.Contact
ADD CONSTRAINT PK_Contact_ContactID
PRIMARY KEY CLUSTERED
(
ContactID ASC
) WITH (PAD_INDEX = ON, FILLFACTOR = 70);
GO
```

- > FILLFACTOR – сколько свободного места на листовом уровне останется для новых данных (чтобы не разделять страниц)
- > PAD_INDEX использует значение, заданное в FILLFACTOR для промежуточных страниц индекса (branch nodes)

Удаление фрагментации

> Перестроение (REBUILD)

- Перестраивается весь индекс
- Операция требует много свободного места в БД
- Выполняется в рамках единой транзакции
 - будьте готовы обеспечить достаточное количество свободного места в логе транзакций

```
ALTER INDEX  
IX_Contact_LastName  
ON Person.Contact  
REBUILD;
```

> Реорганизация (REORGANIZE)

- Сортирует страницы в режиме онлайн
- Требует меньше места в логе транзакций
- Результаты выполнения операции не будут потеряны в случае ошибки/остановки

```
ALTER INDEX IX_Contact_City  
ON Person.Contact  
REORGANIZE;
```

Онлайн операции над индексами

```
ALTER INDEX IX_Contact_EmailAddress  
ON Person.Contact  
REBUILD  
WITH (ONLINE = ON, MAXDOP = 4 );
```

- > Enterprise Edition от SQL Server умеет перестраивать индексы онлайн
- > Обеспечивает одновременный доступ пользователей
- > Медленнее, чем эквивалентная операция в оффлайн режиме
- > По сути, создает новый индекс рядом со старым, поэтому требуется достаточное количество свободного места в файле базы данных

Статистики

Microsoft SQL Server Management Studio

File Edit View Query Project SQL Refactor Debug SQL Prompt Tools Window Community Help

statsdemo.sql - COSIMO\...(\c...an (54))* SQLQuery7.sql - not connected* SQLQuery5.sql - not connected* SQLQuery1.sql - not connected*

```
dbcc show_statistics ('people', stat_lastname)
```

Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows	
1	stat_lastname	Nov 21 2009 11:16AM	200000	147210	200	0.005265445	6.160247	YES	NULL	200000

All density	Average Length	Columns	
1	0.001597444	6.160247	lastname

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS	
1	Abbott	0	200.2286	0	1
2	Adkins	404.8522	1029.554	2	201.456
3	Alexander	657.0328	780.621	1	652.8412
4	Allison	669.3011	409.9275	1	665.0312
5	Andrade	580.6971	202.9344	2	288.9571
6	Arias	617.5018	365.282	3	204.9566
7	Atkins	400.7628	416.692	2	199.4211
8	Avila	410.3047	534.394	2	204.1692
9	Ayers	167.666	695.3886	1	166.5964
10	Baker	592.9653	179.9352	1	589.1824
11	Barajas	657.0328	206.9931	3	218.0775
12	Barker	258.9963	1181.078	1	257.3441
13	Barron	603.8704	420.7507	2	300.4883
14	Bautista	179.9343	389.6341	1	178.7864
15	Becker	791.9836	597.9801	3	262.8694

Query executed successfully. COSIMO\FIRENZE (10.0 SP1) cosimo\christian (54) people 00:00:00 202 rows

Ready Copyright © 2019 Accenture. All rights reserved. Ln 11 Col 46 Ch 46 INS

- Статистика помогает оптимизатору выбрать правильный план выполнения запроса
- Их используют для оценки селективности операции

Обновление статистики

- > Изменение данных приводит к «устареванию» статистики
- > Обновление может быть автоматическим или по запросу
- > AUTO_UPDATE_STATISTICS
 - Опция БД, по умолчанию - ON
- > UPDATE STATISTICS
 - Обновляет статистику оптимизации запросов для таблицы или индексированного представления
- > sp_updatestats
 - Обновляет статистику оптимизации запросов для всей БД
- > ALTER INDEX REBUILD
 - Также обновляет статистику с опцией FULLSCAN

Какие понятия нужно знать

- > Кардинальность колонки (Cardinality) таблицы - число дискретных различных значений колонки, которые встречаются в строках таблицы.
- > Плотность (Density) = Число дубликатов в колонке / Общее число записей в таблице
- > Фактор селективности:

$$selectivity\ factor = \frac{1}{Cardinality}$$

- > Чем меньше фактор селективности, тем меньше требуется операций ввода-вывода для получения результирующего множества строк таблицы. СУБД оценивает эту величину, чтобы решить, применять индекс для доступа к строкам таблицы или нет

Зачем я все это узнал?

Что делать простому разработчику?

Рекомендации по созданию индексов

- > Делайте индексы, покрывающие условие выборки, если данный запрос планируется к использованию достаточно часто.
- > Если есть индекс, уже покрывающий условие выборки в запросе - не создавайте индекс, он лишний
- > Если индекс покрывает почти всё условие запроса - оцените число записей, которое придётся перебрать СУБД при данной выборке. Если оно невелико (менее нескольких тысяч) - не создавайте индекс, он лишний
- > Определите селективность и/или плотность записей в выборке так, как их определит в данном случае оптимизатор. Если в итоговой выборке получится много записей по отношению к общему числу записей в таблице - не создавайте индекс, он лишний

Рекомендации по созданию индексов

- > Когда пишете запрос, думайте о том, как его будет анализировать оптимизатор, сможет ли он корректно посчитать ориентировочное число строк, возвращаемое каждой частью запроса.
- > Колонки, используемые в предложении WHERE, должны быть первыми в составном индексе, иначе он не будет использован. Все последующие колонки должны быть расположены согласно плотности (колонки с более высоким количеством уникальных значений должны быть впереди)
- > В конце концов проверьте план полученного запроса, если он вам действительно важен, особенно если данный запрос нужно выполнять в транзакции.

Рекомендации по выбору колонок для создания индексов

Необходимо помнить о двух основных принципах построения индекса:

- > гарантировать уникальность значений колонки, которая будет индексироваться;
- > увеличить производительность обработки запросов в ХД.

Характеристика колонок, кандидатов для создания индексов	
Хорошие кандидаты	Плохие кандидаты
Колонки Primary Key	Колонки с низкой кардинальностью
Колонки Foreign Key	Колонка имеет много NULL значений
Колонки с уникальными значениями	Колонки с часто изменяемыми значениями
Колонки с операцией соединения	Значительная длина индексных колонок
Колонки с проверяемыми значениями (в WHERE)	
Агрегируемые колонки	

Полезные ссылки

- > Курс [Developing Microsoft SQL Server Databases](#) от MVA
- > [Creating and Optimizing Views in SQL Server](#)
- > [Create Indexed Views](#)
- > [Hypothetical Indexes on SQL Server](#)
- > [CREATE INDEX \(T-SQL\)](#)

Спасибо за внимание!