Индексы

Индекс – это объект базы данных, который представляет собой структуру данных, состоящую из ключей, построенных на основе одного или нескольких столбцов таблицы или представления, и указателей, которые сопоставляются с местом хранения заданных данных. Индексы предназначены для более быстрого получения строк из таблицы, другими словами, индексы обеспечивают быстрый поиск данных в таблице, что значительно повышает производительность запросов и приложений. Индексы также могут быть использованы и для обеспечения уникальности строк таблицы, гарантируя тем самым целостность данных.

Типыиндексовв Microsoft SQL Server

В Microsoft SQL Server существуют следующие типы индексов:

- **Кластеризованный** (*Clustered*) – это индекс, который хранит данные таблицы в отсортированном, по значению ключа индекса, виде. У таблицы может быть только один кластеризованный индекс, так как данные могут быть отсортированы только в одном порядке. По возможности каждая таблица должна иметь кластеризованный индекс, если у таблицы нет кластеризованного индекса, такая таблица называется «*кучей*». Кластеризованный индекс создается автоматически при создании ограничений PRIMARY KEY (*первичный ключ*) и UNIQUE, если до этого кластеризованный индекс для таблицы еще не был определен. В случае создания кластеризованного индекса для таблицы (*кучи*), в которой есть некластеризованные индексы, то после создания все их необходимо перестроить.
- **Некластеризованный** (*Nonclustered*) – это индекс, который содержит значение ключа и указатель на строку данных, содержащую значение этого ключа. У таблицы может быть несколько некластеризованных индексов. Создаваться некластеризованные индексы могут как на таблицах с кластеризованным индексом, так и без него. Именно этот тип индекса используется для повышения производительности часто используемых запросов, так как некластеризованные индексы обеспечивают быстрый поиск и доступ к данным по значениям ключа.

Создание и удаление индексов в Microsoft SQL Server

Перед тем как приступать к созданию индекса его необходимо хорошо спроектировать, для того чтобы эффективно использовать этот индекс, так как плохо спроектированные индексы могут не увеличить производительность, а наоборот снизить ее. Например, большое количество индексов в таблице снижает производительность инструкций INSERT, UPDATE, DELETE, потому что при изменении данных в таблице все индексы должны быть изменены соответствующим образом.

Создание индексов

Для создания индексов в Microsoft SQL Server существует два способа: первый – это с помощью графического интерфейса среды SQL ServerManagementStudio (SSMS), и второй – это с помощью языка Transact-SQL.

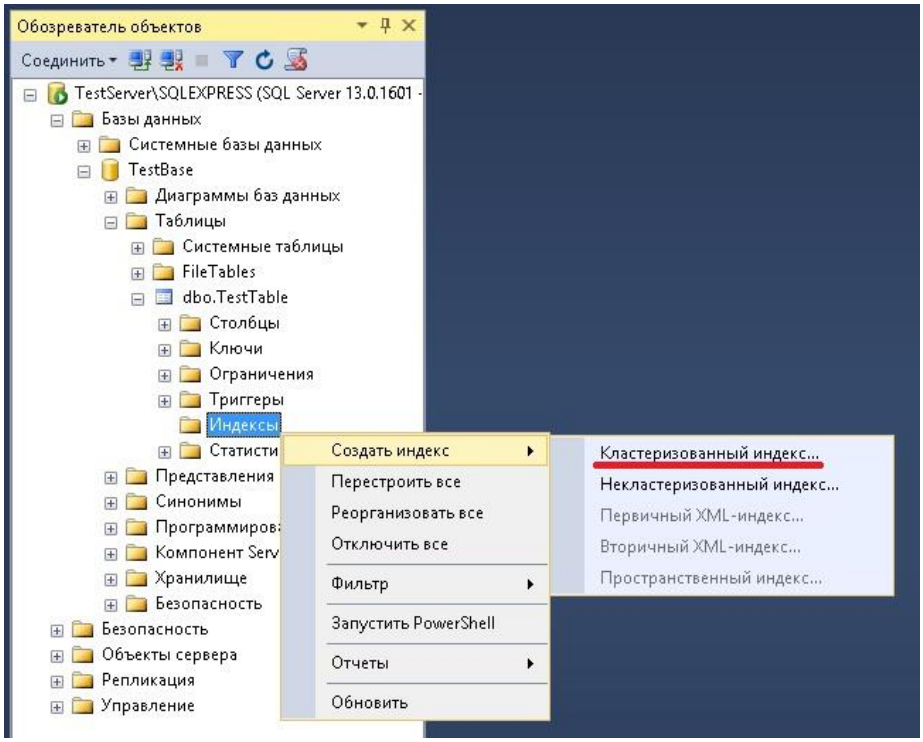
Пример создания кластеризованного индекса

Исходные данные для примера: Таблица с товарами под названием TestTable, в которой есть три столбца: ProductId – идентификатор товара; ProductName – наименование товара; CategoryID – категория товара.

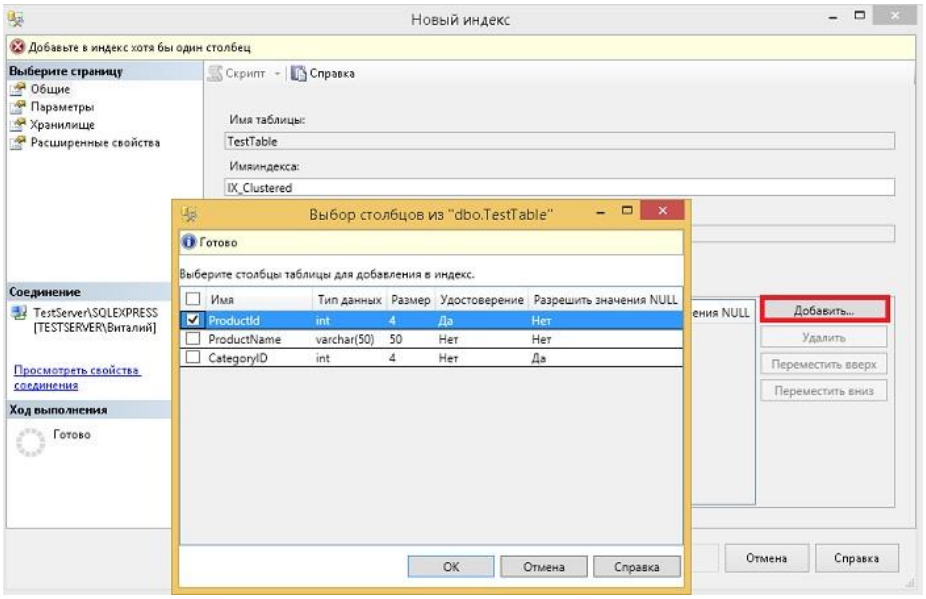
```
CREATE TABLE TestTable(
ProductId INT IDENTITY(1,1) NOT NULL,
ProductNameVARCHAR(50) NOT NULL,
CategoryID INT NULL,
) ON 'PRIMARY'
```

ПримечаниеКластеризованный индекс создается автоматически, если при создании таблицы указать конкретный столбец в качестве первичного ключа (*PRIMARY KEY*).

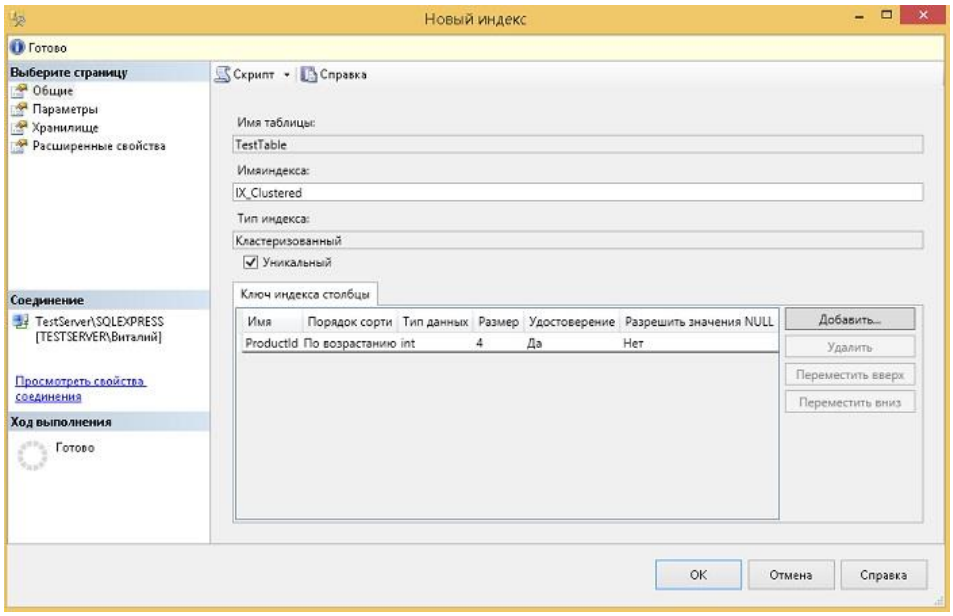
Открываем SSMS и в обозревателе объектов находим нужную таблицу и щелкаем правой кнопкой мыши по пункту «*Индексы*», выбираем «*Создать индекс*» и тип индекса, в нашем случае «*Кластеризованный*».



Откроется форма «*Новый индекс*», где нам необходимо указать имя нового индекса (*оно должно быть уникальным в пределах таблицы*), также указываем, будет ли этот индекс уникальным. Потом выбираем столбец (*ключ индекса*), на основе которого у нас будет создан кластеризованный индекс, т.е. будут отсортированы строки данных в таблице, с помощью кнопки «*Добавить*».



После ввода всех необходимых параметров жмем «*ОК*», в итоге будет создан кластеризованный индекс.



Точно также можно было бы создать кластеризованный индекс, используя инструкцию T-SQL **CREATRE INDEX**, например, воттак

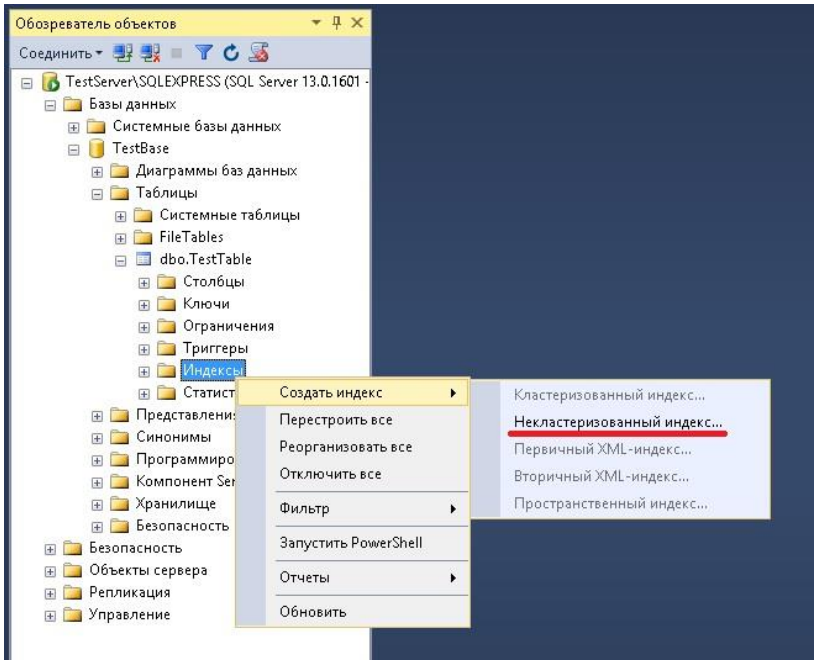
```
    CREATE UNIQUE CLUSTERED INDEX IX_Clustered ON TestTable
(
ProductId ASC
  )
  GO
```

Пример создания некластеризованного индекса с включенными столбцами

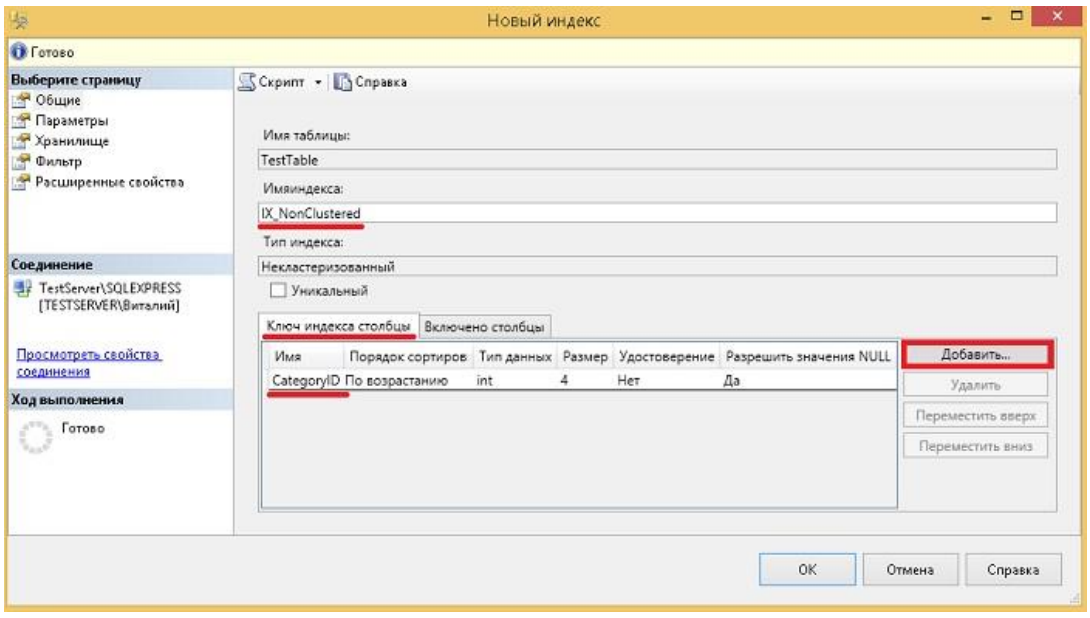
Рассмотрим пример создания некластеризованного индекса, при этом укажем столбцы, которые не будет являться ключевыми, но которые будут включаться в индекс. Это полезно в тех случаях, когда создается индекс для конкретного запроса, например, для того чтобы индекс полностью покрывал запрос, т.е. содержал все столбцы (*это*

называется «Покрытием запроса»). Благодаря покрытию запроса повышается производительность, так как оптимизатор запросов может найти все значения столбцов в индексе, при этом не обращаясь к данным таблиц, что приводит к меньшему числу дисковых операций ввода-вывода. Однако включение в индекс неключевых столбцов влечет за собой увеличение размера индекса, т.е. для хранения индекса потребуется больше места на диске, а также может повлечь и снижение производительности операций INSERT, UPDATE, DELETE на базовой таблице.

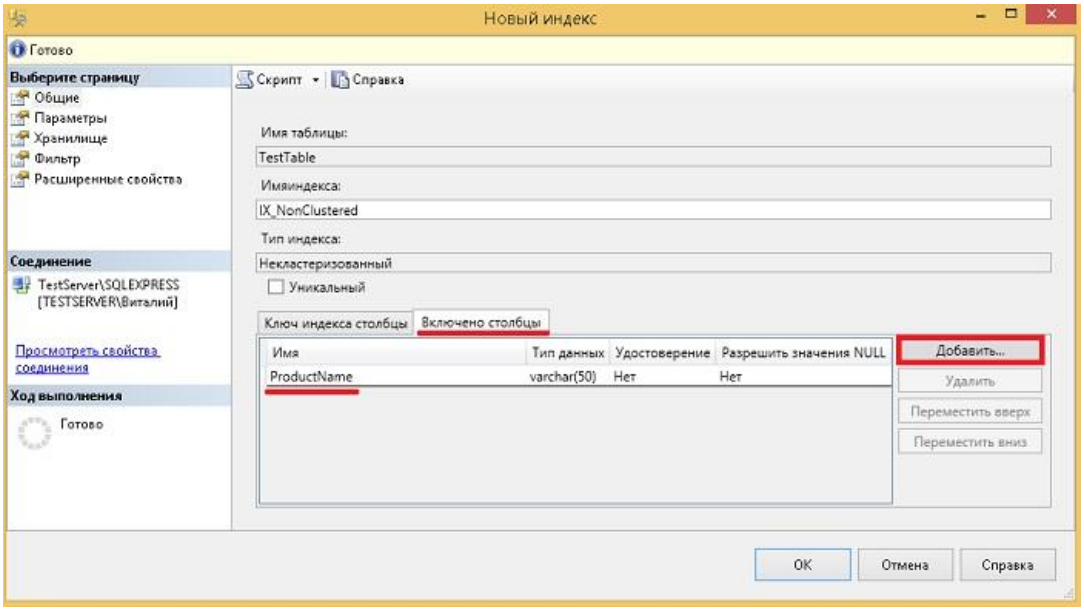
Для того чтобы создать некластеризованный индекс с помощью графического интерфейса ManagementStudio, мы также находим нужную таблицу, выбираем пункт Индексы, затем «*Создать ->Некластеризованный индекс*».



После открытия формы «*Новый индекс*» указываем название индекса, добавляем ключевой столбец или столбцы с помощью кнопки «*Добавить*», например, столбец CategoryID.



Далее переходим на вкладку «*Включено столбцы*» и с помощью кнопки «*Добавить*» добавляем столбцы, которые необходимо включить в индекс, например, ProductName.

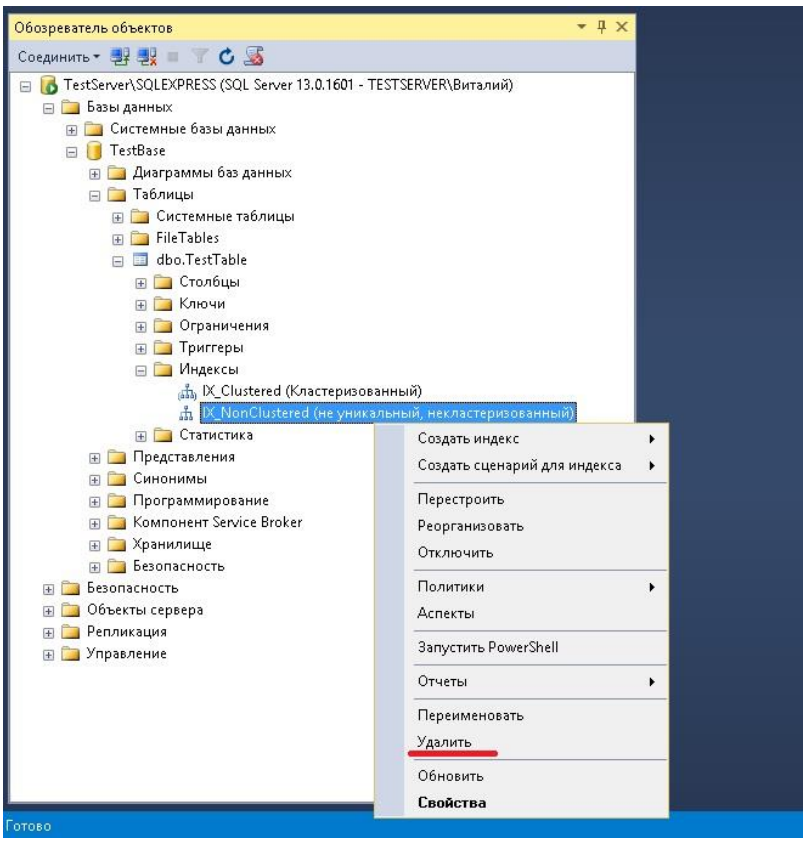


На Transact-SQL это будет выглядеть следующим образом.

```
CREATE NONCLUSTERED INDEX IX_NonClustered ON TestTable
  (
CategoryID ASC
  )
  INCLUDE (ProductName)
  GO
```

Пример удаления индекса в Microsoft SQL Server

Для того чтобы удалить индекс можно щелкнуть правой кнопкой по нужному индексу и нажать «*Удалить*», затем подтвердить свое действия нажав «*ОК*».



или также можно использовать инструкцию **DROP INDEX**, например

```
DROP INDEX IX_NonClustered ON TestTable
```

Следует отметить, что инструкция DROP INDEX неприменима к индексам, которые были созданы путем создания ограничений PRIMARY KEY и UNIQUE. В данном случае для удаления индекса нужно использовать инструкцию ALTER TABLE с предложением DROP CONSTRAINT.

Оптимизация индексов в Microsoft SQL Server

В результате выполнения операций обновления, добавления или удаления данных в таблицах SQL сервер автоматически вносит соответствующие изменения в индексы, но со временем все эти изменения могут вызвать фрагментацию данных в индексе, т.е. они окажутся разбросанными по базе данных. Фрагментация индексов влечет за собой снижение производительности запросов, поэтому периодически необходимо выполнять операции обслуживания индексов, а именно дефрагментацию, к таким можно отнести операции реорганизации и перестроения индексов.

В каких случаях использовать реорганизацию индекса, а в каких перестроение?

Чтобы ответить на этот вопрос сначала необходимо определить степень фрагментации индекса, так как в зависимости от фрагментации индекса тот или иной метод дефрагментации будет предпочтительней и эффективней.

Для получения информации о внутренней фрагментации индекса применяется динамическое административное представление DMV, называемое

sys.dm_db_index_physical_stats. Это DMV возвращает информацию об объеме и фрагментации данных и индексов указанной страницы. Для каждой страницы возвращается одна строка для каждого уровня B+-дерева.

```
USE TestBase;
DECLARE @dbId INT;
DECLARE @tabId INT;
DECLARE @indId INT;
SET @dbId = DB_ID('TestBase');
SET @tabId = OBJECT_ID('TestTable');
SELECT avg_fragmentation_in_percent, avg_page_space_used_in_percent
FROM sys.dm_db_index_physical_stats (@dbId, @tabId, NULL, NULL, NULL);
```

Представление sys.dm_db_index_physical_stats имеет пять параметров: идентификатор текущей базы данных, таблицы и индекса соответственно; идентификатор раздела, уровень сканирования, применяемый для получения статистической информации (значение по умолчанию для определенного параметра можно указать посредством значения NULL). Наиболее важными из столбцов этого представления являются столбцы avg_fragmentation_in_percent и avg_page_space_used_in_percent. В первом указывается средний уровень фрагментации в процентах, а во втором определяется объем занятого пространства в процентах.

Microsoft рекомендует:

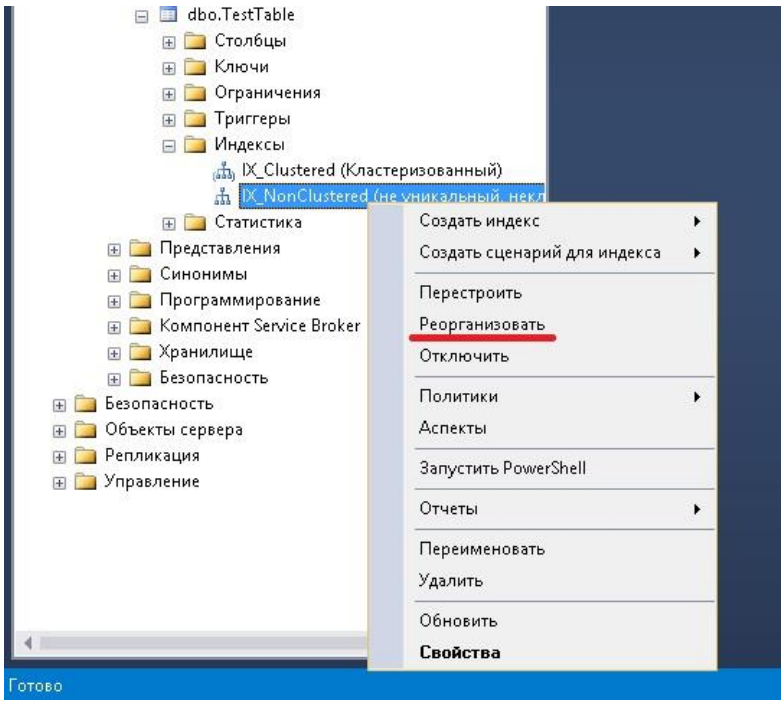
- Если степень фрагментации менее 5%, то реорганизацию или перестроение индекса вообще не стоит запускать;
- Если степень фрагментации от 5 до 30%, то имеет смысл запустить реорганизацию индекса, так как данная операция использует минимальные системные ресурсы и не требует долговременных блокировок;
- Если степень фрагментации более 30%, то необходимо выполнять перестроение индекса, так как данная операция, при значительной фрагментации, дает больший эффект чем операция реорганизации индекса.

Реорганизация индексов

Реорганизация индекса – это процесс дефрагментации индекса, который дефрагментирует конечный уровень кластеризованных и некластеризованных индексов по таблицам и представлениям, физически переупорядочивая страницы концевого уровня в соответствии с логическим порядком (*слева направо*) конечных узлов.

Для реорганизации индекса можно использовать как графический инструмент SSMS, так и инструкцию Transact-SQL.

Реорганизация индекса с помощью ManagementStudio



Реорганизация индекса с помощью Transact-SQL

```
ALTER INDEX IX_NonClustered ON TestTable
REORGANIZE
  GO
```

Перестроение индексов

Перестроение индекса – это процесс, при котором происходит удаление старого индекса и создание нового, в результате чего фрагментация устраняется.

Для перестроения индексов можно использовать два способа.

Первый. Используя инструкцию ALTER INDEX с предложением REBUILD. Обычно для массового перестроения индексов используется именно этот способ.

Пример

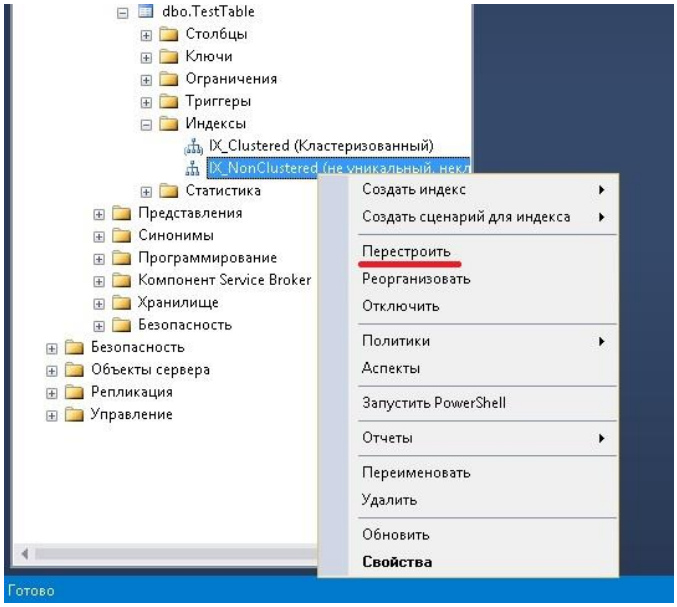
```
ALTERINDEXIX_NonClusteredONTestTable
REBUILD
  GO
```

И второй, используя инструкцию CREATEINDEXспредложениемDROP_EXISTING. Можно использовать, например, для перестроения индекса с изменением его определения, т.е. добавления или удаления ключевых столбцов.

Пример

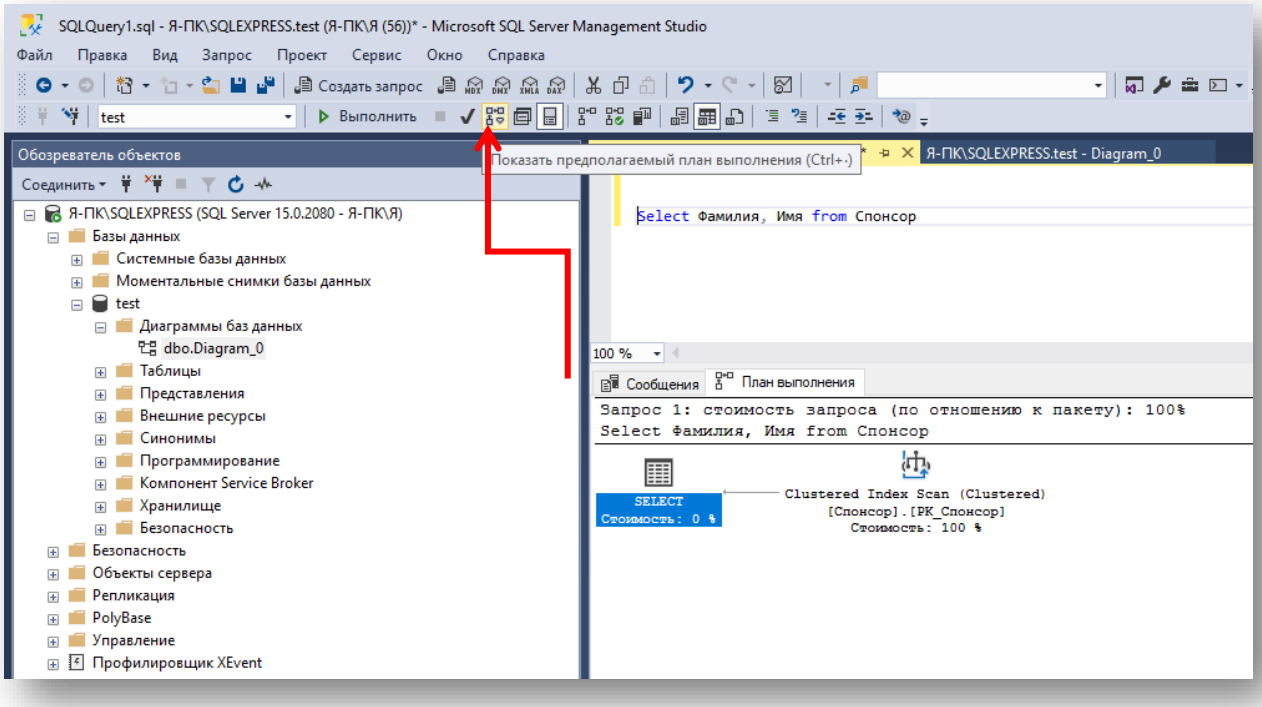
```
    CREATE NONCLUSTERED INDEX IX_NonClustered ON TestTable
  (
CategoryID ASC
  )
WITH(DROP_EXISTING = ON)
GO
```

В ManagementStudio функционал для перестроения также доступен. Правой кнопкой по нужному индексу «*Перестроить*».

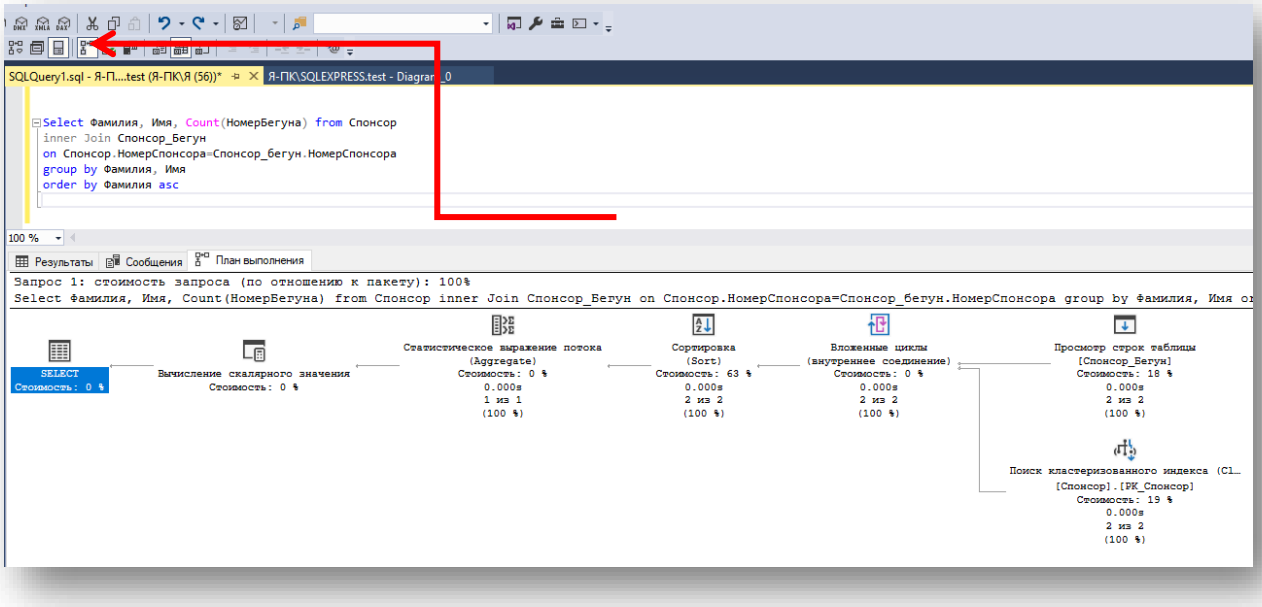


Просмотр предполагаемого плана выполнения запроса

Для упрощения оптимизации запросов в ManagementStudio в редактор кода встроен функционал, который позволяет посмотреть план выполнения запросов, и в случае если он не оптимален, он предложит, например, создать тот или иной индекс.



Действительный план выполнения запроса



Действительный план выполнения аналогичен предполагаемому плану выполнения, но включает также действительные (не предполагаемые) значения количества строк, количества перемоток и т. д.
Чтобы включить в запрос действительный план выполнения, необходимо выбрать из меню Query (Запрос) команду IncludeActualExecutionPlan (Включить действительный план выполнения). Затем необходимо нажать F5 и выполнить запрос.

Индексированные представления

Компонент DatabaseEngine позволяет создавать индексы для представлений. Такие представления называются индексированными представлениями.
Чтобы создать представление индексированным, необходимо создать **кластеризованный** индекс для столбца (столбцов) этого представления (кластеризованный индекс является единственным типом индекса, который содержит значения данных в своих страницах узлов).
После создания такого индекса система баз данных выделяет память для этого представления, после чего можно создавать любое число **некластеризованных** индексов, поскольку теперь это представление рассматривается как базовая таблица.

Условия создания индексированных представлений

Индексированное представление можно создать только в том случае, если оно является детерминированным, т.е. представление всегда возвращает один и тот же результирующий набор. Для этого следующим параметрам инструкции SET нужно присвоить значение ON:

```
QUOTED_IDENTIFIER
CONCAT_NULL_YIELDS_NULL
ANSI_NULLS
ANSI_PADDING
ANSI_WARNINGS
```

Кроме этого, параметру `NUMERIC_ROUNDABORT` нужно присвоить значение off.

Пример создания индексированных представлений

```
--Установка опций сервера (см. выше)

--Создание представления с schemabinding
CREATEVIEW V1Заказ
WITH SCHEMABINDING
AS
SELECT SUM(ЦенаЗаЕд*Количество)ASДоход,
ДатаЗаказа,НомерПродукта
FROM dbo.Детализация, dbo.Заказы
WHERE dbo.Заказы.НомерЗаказа = dbo.Детализация.НомерЗаказа
GROUPBY ДатаЗаказа, НомерПродукта;
GO

--Создание кластеризованногоиндекса на представлении
CREATEUNIQUECLUSTEREDINDEX I_V1Заказ
ON  V1Заказ(ДатаЗаказа, НомерПродукта);
GO
```

Оптимизатор запросов и индексированные представления

Обычно для выполнения запроса выбирается индекс таблицы, а не представления. Это связано с тем, что

- извлечение данных из базовой таблицы дешевле извлечения данных из индекса представления. Разница в скорости извлечении данных из представления и из базовой таблицы составит доли миллисекунд, если не меньше. Но она, разница эта – есть!
- оптимизатор может оценить таблицу из которой реально идет извлечение данных как «небольшую» (*smalltable*). Если такое решение принято, дальнейшее общение идет только с ней, оптимизатор начисто игнорирует представление и все его индексы.
- оптимизатор может оценить таблицу из которой реально идет извлечение данных как «нормальную» (то есть не *small*), однако при этом оценить сам запрос эти данные извлекающий как тривиальный (*trivial*). Результат такой оценки полностью эквивалентен предыдущему пункту – работа продолжается только с таблицей/табличными индексами, и ни в коем случае не с представлением/индексами на нем.

Если созданные индексы не используются оптимизатором запросов, то следует указать явно, на необходимость их использования, добавив хинт `WITH(NOEXPAND)`.

NOEXPAND указывает, что при обработке запроса оптимизатором запросов никакие индексированные представления не расширяются для доступа к базовым таблицам. Оптимизатор запросов обрабатывает представление так же, как и таблицу с кластеризованным индексом.

Пример:

```
SELECT Доход,ДатаЗаказа,НомерПродукта FROM V1Заказ WITH(NOEXPAND)WHERE
Доход>1000
```

Очень подробно вопрос «Почему не используются индексы, созданные на представлении, оптимизатором запросов» рассмотрен в статье [http://sqlcmd.ru/indexed_views_workings-part02.html.](http://sqlcmd.ru/indexed_views_workings-part02.html)

Задания

1. Создать «покрывающий» некластеризованный индекс для повышения производительности запросов.

1.1. Запросы придумать самостоятельно. Количество запросов/индексов – 4 шт.

1.2. Просмотреть предполагаемый и действительный план выполнения каждого запроса.

1.3. Созданные индексы используются при выполнении запросов? Если нет, то удалить индекс и создать другой.

2. Создать индексированное представление ViewGoToExam, предоставляющее доступ ко всем студентам, сдававшим экзамены.

Замечание: Нужно включить в представление поле/поля для создания ключа кластеризованного индекса (например, поле КодУспеваемости). Поле или совокупность полей должны содержать уникальные значения.

2.1 При создании индексированного представления обязательно установить опции сервера и использовать SCHEMABINDING.

2.2 Создать кластеризованный индекс для столбца (столбцов) этого представления.

2.3 Создать некластеризованный индекс для повышения производительности запроса:

```
SELECT ФамилияСтудента, ИмяСтудента, ОтчествоСтудента, Оценка, НазваниеПредмета FROM ViewGoToExam WHERE ДатаОценки BETWEEN '01.01.2024' AND '01.06.2024';
```

2.4 Создать некластеризованный индекс для повышения производительности запроса:

```
SELECT ФамилияСтудента, ИмяСтудента, ОтчествоСтудента, AVG(Оценка) as [Средний балл] FROM ViewGoToExam GROUP BY ФамилияСтудента, ИмяСтудента, ОтчествоСтудента;
```

2.5 Проверить использование созданных индексов при выполнении запросов.

3. Оценить фрагментацию индексов в любой таблице с некластеризованными индексами. Нужно ли перестраивать или реорганизовывать индексы?