**Цели работы:**

* изучить назначение и разновидности индексов;
* научиться создавать, удалять, перестраивать и реорганизовывать индексы средствами MS SQL Server Management Studio, а также средствами языка T-SQL;
* изучить возможности операторов T-SQL для работы с индексами.

**Теоретическая часть:**

Одним из главных методов оптимизации запросов на выборку является добавление индексов к таблице.

**Индекс** – это объект базы данных, который представляет собой структуру данных, состоящую из ключей, построенных на основе одного или нескольких столбцов таблицы, и указателей, которые сопоставляются с местом фактического хранения заданных данных. Иными словами, индексы – это некие ссылки на данные, своего рода предметный указатель, по которым можно быстро найти и получить фактические данные, т.е. строки таблицы. Основное назначение индексов - это как раз быстрый поиск необходимых строк таблицы.

Во время выполнения запроса оптимизатор (это встроенный механизм в SQL сервере, который отвечает за разбор и превращение текста запроса в конкретные действия) выбирает оптимальный и самый быстрый план выполнения запроса, т.е. получение данных. Если у таблицы есть индексы, оптимизатор обязательно выберет план, в котором задействуется индекс, а если индексов несколько, то он будет выбирать наиболее эффективный индекс для конкретного запроса. С увеличением столбцов в индексе, размер индекса также растет.

**Типы индексов**

В Microsoft SQL Server существуют несколько типов 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»;
* **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;
* **фильтруемый** (Filtered) – это оптимизированный некластеризованный индекс, который использует предикат фильтра для индексирования части строк в таблице. Если хорошо спроектировать такой тип индекса, то он может повысить производительность запросов, а также снизить затраты на обслуживание и хранение индексов по сравнению с полнотабличными индексами;
* **уникальный** (Unique) – это индекс, который обеспечивает отсутствие повторяющихся (одинаковых) значений ключа индекса, гарантируя тем самым уникальность строк по данному ключу. Уникальными могут быть как кластеризованные, так и некластеризованные индексы. Если создавать уникальный индекс по нескольким столбцам, индекс гарантирует уникальность каждой комбинации значений в ключе. Уникальный индекс может быть создан только в том случае, если у таблицы на текущий момент отсутствуют дублирующие значения по ключевым столбцам;
* **колоночный** (Columnstore) – это индекс, основанный на технологии хранения данных в виде столбцов. Данный тип индекса эффективно использовать для больших хранилищ данных, поскольку он может увеличить производительность запросов к хранилищу до 10 раз и также до 10 раз уменьшить размер данных, так как данные в Columnstore индексе сжимаются. Существуют как кластеризованные колоночные индексы, так и некластеризованные;
* **полнотекстовый** (Full-text) – это специальный тип индекса, который обеспечивает эффективную поддержку сложных операций поиска слов в символьных строковых данных;
* **пространственный** (Spatial) – это индекс, который обеспечивает возможность более эффективного использования конкретных операций на пространственных объектах в столбцах с типом данных geometry или geography. Данный тип индекса может быть создан только для пространственного столбца;
* **XML** – это еще один специальный тип индекса, который предназначен для столбцов с типом данных XML. Благодаря XML-индексу повышается эффективность обработки запросов к XML столбцам;
* также существуют специальные индексы для таблиц, оптимизированных для памяти (In-Memory OLTP), такие как: Хэш (Hash) индексы и некластеризованные индексы, оптимизированные для памяти, которые создаются для сканирования диапазона и упорядоченного сканирования.

Кластеризованный, некластеризованный и уникальный индексы имеются практически во всех базах данных, так как они просто жизненно необходимы, остальные типы индексов уже более специфичны и их наличие зависит от того, какие данные хранит база данных, т.е. для чего она предназначена.

Кластеризованный индекс – это один из основных индексов и, можно сказать, обязательный тип индекса, который должен быть у таблицы. Кластеризованный индекс даже создается автоматически, если, например, при создании таблицы указан конкретный столбец в качестве первичного ключа (PRIMARY KEY).

**Создание индексов:**

В T-SQL индексы создаются с помощью инструкции CREATE.

Базовый синтаксис:

CREATE [CLUSTERED | NONCLUSTERED | UNIQUE] INDEX index\_name

ON [database\_name.[schema\_name].|schema\_name.] table\_name

({column1 [ASC | DESC]} [,… n])

[INCLUDE (column1)]

[WITH (DROP\_EXISTING = {ON | OFF})]

#### где

#### index\_name – имя создаваемого индекса. Имена индексов должны быть уникальными в пределах таблицы или представления, но их уникальность не обязательна в пределах базы данных. Имена индексов должны удовлетворять правилам для идентификаторов.

#### table\_name – имя таблицы, в которой будет создан индекс.

#### UNIQUE – создает уникальный индекс для таблицы или представления. Уникальным является индекс, в котором не может быть двух строк с одним и тем же значением ключа индекса. Кластеризованный индекс представления должен быть уникальным.

#### CLUSTERED – создает индекс, в котором логический порядок значений ключа определяет физический порядок соответствующих строк в таблице. На нижнем (конечном) уровне кластеризованного индекса хранятся действительные строки данных таблицы. Для таблицы или представления в каждый момент времени может существовать только один кластеризованный индекс.

#### NONCLUSTERED – создает индекс, задающий логическое упорядочение для таблицы. Логический порядок строк в некластеризованном индексе не влияет на их физический порядок. Каждая таблица может содержать до 999 некластеризованных индексов независимо от способа их создания: неявно с помощью ограничений PRIMARY KEY и UNIQUE или явно с помощью инструкции CREATE INDEX.

#### column – столбец или столбцы, на которых основан индекс. Имена одного или нескольких столбцов для создания комбинированного индекса. В один ключ составного индекса могут входить до 32 столбцов.

#### ASC | DESC – определяет сортировку значений заданного столбца индекса: по возрастанию или по убыванию. Значение по умолчанию — ****ASC****.

#### INCLUDE (column [ ,... n ] ) – указывает неключевые столбцы, добавляемые на конечный уровень некластеризованного индекса. Некластеризованный индекс может быть уникальным или неуникальным.

#### DROP\_EXISTING = { ON | OFF } – параметр для удаления и перестроения существующего кластеризованного или некластеризованного индекса с измененными спецификациями столбцов и сохранения того же имени для индекса. Значение по умолчанию — **OFF**.

Примеры:

1. Создание некластеризованного индекса в таблице или представлении.

CREATE INDEX index1

ON schema1.table1 (column1)

1. Создание кластеризованного индекса в таблице и использование имени, состоящего из трех элементов, для таблицы.

CREATE CLUSTERED INDEX index1

ON database1.schema1.table1 (column1)

1. Создание некластеризованного индекса с ограничением уникальности и указание порядка сортировки.

CREATE UNIQUE INDEX index1

ON schema1.table1

(

column1 DESC,

column2 ASC,

column3 DESC

)

1. Создание некластеризованного индекса со включенными столбцами.

CREATE NONCLUSTERED INDEX IX\_NonClustered

ON TestTable

(

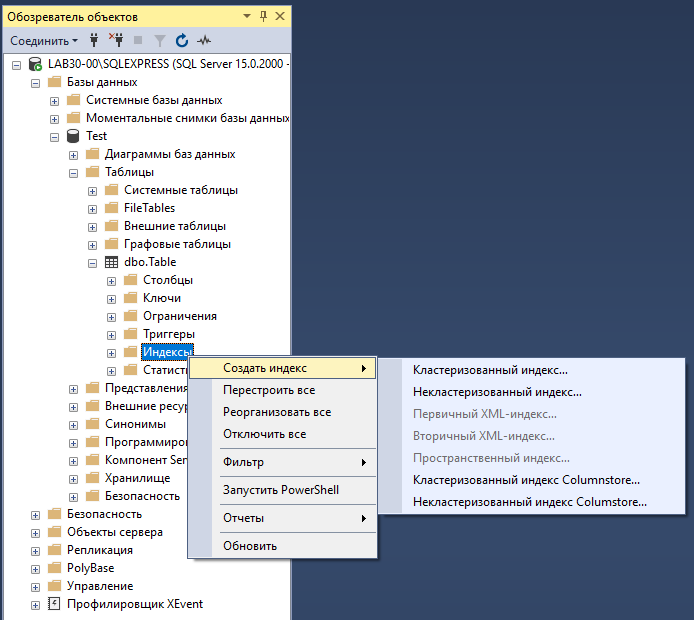
CategoryId ASC,

ProductName ASC

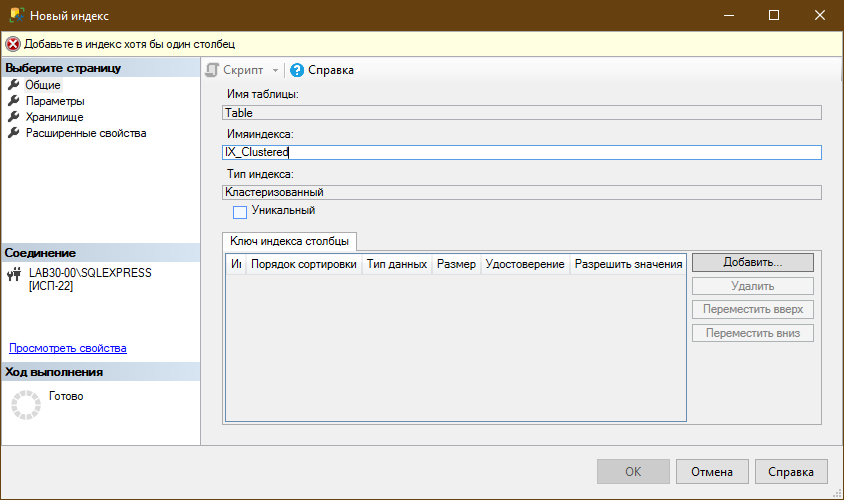
)

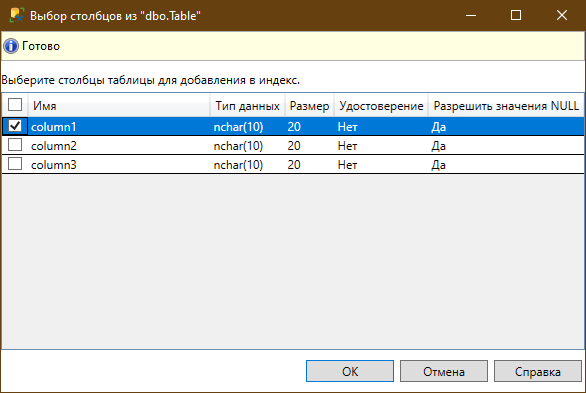
INCLUDE (Price)

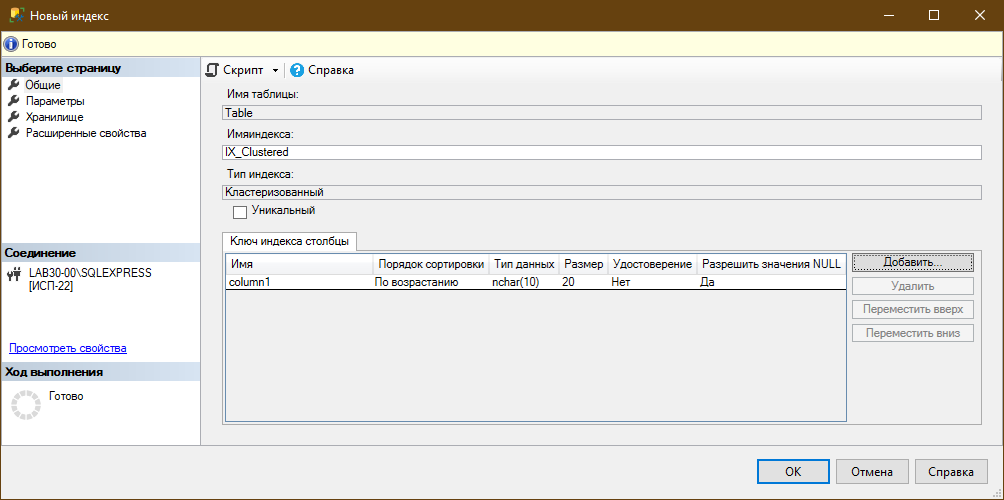
Индексы можно создавать также и с помощью SQL Server Management Studio, для этого в обозревателе объектов находим нужную таблицу и щелкаем правой кнопкой мыши по пункту «Индексы», выбираем «Создать индекс» и тип индекса.



Откроется форма «Новый индекс», где необходимо указать имя нового индекса, а также настроить его. Потом с помощью кнопки «Добавить», выбрать столбцы (ключ индекса), на основе которых будет создан индекс.







Принудительно (т.е. как показано выше) создается кластеризованный индекс редко, так как он автоматически создается, если указан первичный ключ, а первичный ключ необходим практически для большинства таблиц.

Гораздо чаще будет создаваться некластеризованные индексы, которые как раз и используют для более быстрого доступа к данным.

**Удаление и изменение индексов:**

Индексы созданы, но иногда требуется изменить эти индексы, например, добавить еще один ключевой столбец или добавить так называемые «**Включенные столбцы**» - это столбцы, которые не являются ключевыми, но включаются в индекс. За счет этого уменьшается количество дисковых операций ввода-вывода и скорость доступа к данным, соответственно, увеличивается.

Для внесения изменений нужно сначала удалить индекс, это делается следующим образом (все действия также можно сделать и в Management Studio), затем пересоздать.

Базовый синтаксис:

DROP INDEX [IF EXISTS] IX\_Name

ON tableName

**IF EXISTS** – условное удаление индекса только в том случае, если он уже существует.

**IX\_Name** – имя индекса, который необходимо удалить.

Например:

DROP INDEX IX\_NonClustered

ON TestTable

Синтаксис T-SQL позволяет в одной инструкции и удалить индекс, и создать его заново, например, для того чтобы изменить его определение. Параметр DROP\_EXISTING = ON, что означает, что индекс существует и его необходимо удалить и создать заново.

CREATE NONCLUSTERED INDEX IX\_NonClustered

ON TestTable

(

CategoryId ASC,

ProductName ASC

)

INCLUDE (Price)

WITH (DROP\_EXISTING = ON)

**Проектирование индексов:**

Рекомендации по проектированию индексов:

* одним из самых эффективных индексов является индекс для целочисленных столбцов, которые имеют уникальные значения, поэтому по возможности создавайте индексы для таких столбцов;
* если таблица очень интенсивно обновляется, то не рекомендуется создавать большое количество индексов, так как это снижает производительность инструкций INSERT, UPDATE, DELETE и MERGE. Потому что после изменений данных в таблице, SQL сервер автоматически вносит соответствующие изменения во все индексы;
* если таблица с большим объемом данных обновляется редко, при этом она активно используется в инструкциях SELECT, т.е. на выборку данных, то большое количество индексов может улучшить производительность, так как у оптимизатора запросов будет больший выбор индексов при определении наиболее эффективного и быстрого способа доступа к данным;
* для таблиц с небольшим объемом данных создание некластеризованных индексов с целью повышения производительности может оказаться абсолютно бесполезно, да еще и с затратами на их поддержание. Так как оптимизатору может потребоваться больше времени на поиск данных в индексе, чем просмотр данных в самой таблице. Поэтому не создавайте индексы для таблиц, в которых очень мало данных;
* кластеризованный индекс необходимо создавать для столбца, который является уникальным и не принимает значения NULL, также длина ключа должна быть небольшой, другими словами, ключ индекса не нужно составлять из нескольких столбцов;
* некластеризованные индексы лучше всего создавать для всех столбцов, которые часто используются в условиях (WHERE) и в объединениях (JOIN);
* по возможности не стоит создавать индексы, в которых очень много ключевых столбцов, так как это влияет на размер индекса и на ресурсы его поддержания;
* эффективно использовать покрывающие индексы, т.е. индексы которые включают все столбцы, используемые в запросе. Благодаря этому 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. Это можно достичь с помощью включения в индекс неключевых столбцов (включенные столбцы), но также следует принять во внимание, что это влечет за собой увеличение размера индекса;
* если есть возможность, то рекомендовано заменять неуникальный индекс уникальным для той же комбинации столбцов, это обеспечивает оптимизатору запросов дополнительные сведения, что может сделать индекс более эффективным;
* при создании индекса учитывайте порядок ключевых столбцов, это повышает производительность индекса. Например, столбцы, которые используются в предложении WHERE в условиях поиска равно (=), больше (>), меньше (<) или находящихся в интервале (BETWEEN) или участвуют в соединении (JOIN), должны стоять первыми. Если таких несколько, то упорядочивайте их по уровню различности, т.е. от наиболее четкого к наименее четкому.

**Обслуживание индексов:**

Индексы спроектированы и созданы, и у нас запросы работают быстро, но в результате выполнения операций обновления, добавления или удаления данных в таблицах SQL сервер автоматически вносит соответствующие изменения в индексы, и 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.

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

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

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 определения степени фрагментации индекса можно использовать системную табличную функцию **sys.dm\_db\_index\_physical\_stats** (что такое табличные функции, мы рассмотрим чуть позже), которая возвращает подробные сведения о размере и фрагментации индексов.

Например, используя следующий запрос, можно узнать степень фрагментации индексов у всех таблиц в текущей базе данных.

SELECT OBJECT\_NAME(T1.object\_id) AS NameTable,

T1.index\_id AS IndexId,

T2.name AS IndexName,

T1.avg\_fragmentation\_in\_percent AS Fragmentation

FROM sys.dm\_db\_index\_physical\_stats (DB\_ID(), NULL, NULL, NULL, NULL) AS T1

LEFT JOIN sys.indexes AS T2 ON T1.object\_id = T2.object\_id

AND T1.index\_id = T2.index\_id

В данном случае нас интересует столбец **avg\_fragmentation\_in\_percent**, т.е. процентная доля логической фрагментации. 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 индекса:**

ALTER INDEX IX\_NonClustered

ON TestTable

REORGANIZE

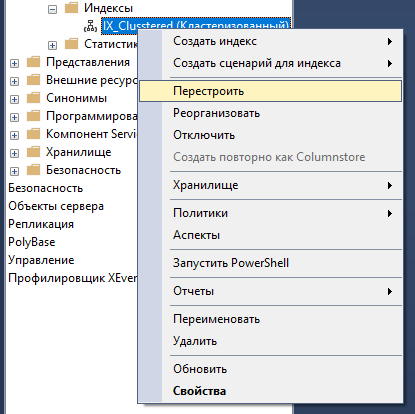
**Перестроение индекса:**

ALTER INDEX IX\_NonClustered

ON TestTable

REBUILD

Средствами SQL Server Management Studio сделать это можно, выбрав соответствующий пункт контекстного меню для нужного индекса.



В процессе выполнения инструкции CREATE INDEX с предложением DROP\_EXISTING, если вносятся изменения в его определение, происходит перестроение индекса.

**Задание:**

* 1. Запустить SQL Server Management Studio, выполнить подключение к серверу (используя параметры соединения, аналогичные практической 1).
  2. Для созданных в практической работе 1 таблиц добавить необходимые индексы (выбор обосновать).

**В отчет предоставить:**

* текст SQL-запроса, выполняющий добавление индексов;
* скриншот структуры БД, показывающий добавленные индексы (на скриншоте обязательно должно присутствовать имя (номер) компьютера).

**Контрольные вопросы:**

1. Опишите назначение индекса.
2. За что отвечает кластеризованный индекс?
3. За что отвечает некластеризованный индекс?
4. Сколько может быть кластеризованных и некластеризованных индексов у таблицы?
5. Что представляет собой уникальный индекс?
6. Опишите базовый синтаксис оператора CREATE INDEX.
7. Какие рекомендации по проектированию индексов существуют?
8. Для чего нужна реорганизация индекса?
9. Для чего нужна перестройка индекса?
10. В каких случаях проводится реорганизация, а в каких перестройка?

**Порядок оценивания:**

Оценка «**Отлично**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 5 вопросов.

Оценка «**Хорошо**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 4 вопроса.

Оценка «**Удовлетворительно**» выставляется, если были выполнены все пункты задания, отчёт содержит минимально необходимую информацию по проделанной работе, работа сдана и защищена с нарушением сроков, в ходе защиты даны правильные ответы на 2-3 вопроса.

Оценка «**Неудовлетворительно**» выставляется, если были выполнены не все пункты задания, отчёт содержит информацию, не позволяющую оценить ход выполнения работы, работа сдана с нарушением сроков, не защищена, либо в ходе защиты даны правильные ответы менее чем на 2 вопроса.

Работы, получившие оценку «Удовлетворительно» и выше являются засчитанными.

Работы, получившие оценку «Неудовлетворительно» являются обязательными к доработке и повторной сдаче. Итоговая оценка при повторной сдаче работы будет снижена на 1 балл.