Каждая страница размером 8192 байт включает заголовок, имеющий длину 96 байт. Еще один фрагмент страницы используется для размещения других структур данных, например, информации о переполнении строк. Вся оставшаяся часть страницы (8060 байт) предназначена для размещения данных (т.е. информации таблицы или индекса).

Индексы создаются для столбцов таблиц и представлений. Индексы предоставляют путь для быстрого поиска данных на основе значений в этих столбцах.

Индекс состоит из набора страниц, узлов индекса, которые организованы в виде древовидной структуры — *сбалансированного дерева*. Эта структура является иерархической по своей природе и начинается с корневого узла на вершине иерархии и конечных узлов, листьев, в нижней части, как показано на рисунке:

Когда вы формируете запрос на индексированный столбец, подсистема запросов начинает идти сверху от корневого узла и постепенно двигается вниз через промежуточные узлы, при этом каждый слой промежуточного уровня содержит более детальную информацию о данных. Подсистема запросов продолжает двигаться по узлам индекса до тех пор, пока не достигнет нижнего уровня с листьями индекса.

Индексы создаются автоматически при определении ограничений PRIMARY KEY или UNIQUE на основе столбцов таблицы. Например, при создании таблицы и указании конкретного столбца в качестве первичного ключа, компонент Компонент Database Engine автоматически создает на основе этого столбца ограничение PRIMARY KEY и индекс. Дополнительные сведения см. в разделах [Create Primary Keys](https://msdn.microsoft.com/ru-ru/library/ms189039.aspx) и [Create Unique Constraints](https://msdn.microsoft.com/ru-ru/library/ms190024.aspx).

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.

Компонент Компонент Database Engine не позволяет создать уникальный индекс по столбцам, уже содержащим повторяющиеся значения, даже если параметру IGNORE\_DUP\_KEY присвоено значение ON. При попытке создания такого индекса компонент Компонент Database Engine выдает сообщение об ошибке. Прежде чем создавать уникальный индекс по такому столбцу или столбцам, необходимо удалить все повторяющиеся значения. Столбцы, используемые в уникальном индексе, должны иметь свойство NOT NULL, т. к. при создании индекса значения NULL рассматриваются как повторяющиеся.

Инструкция DROP INDEX неприменима к индексам, созданным при указании ограничений параметров PRIMARY KEY и UNIQUE. Для удаления ограничения и соответствующего индекса используется [ALTER TABLE](https://msdn.microsoft.com/ru-ru/library/ms190273.aspx) с предложением DROP CONSTRAINT.

**Кластеризованный индекс**

Кластеризованный индекс хранит реальные строки данных в листьях индекса. Важной характеристикой кластеризованного индекса является то, что все значения отсортированы в определенном порядке либо возрастания, либо убывания. Таким образом, таблица или представление может иметь только один кластеризованный индекс. В дополнение следует отметить, что данные в таблице хранятся в отсортированном виде только в случае если создан кластеризованный индекс у этой таблицы.  
Таблица не имеющая кластеризованного индекса называется кучей.

Кластеризованный индекс создается по умолчанию для каждой таблицы, для которой с помощью ограничения первичного ключа определен первичный ключ. Кроме этого, каждый кластеризованный индекс однозначен по умолчанию, т.е. в столбце, для которого определен кластеризованный индекс, каждое значение данных может встречаться только один раз. Если кластеризованный индекс создается для столбца, содержащего повторяющиеся значения, система баз данных принудительно обеспечивает однозначность, добавляя четырехбайтовый идентификатор к строкам, содержащим дубликаты значений.

**Некластеризованный индекс**

В отличие от кластеризованного индекса, листья некластеризованного индекса содержат только те столбцы (*ключевые*), по которым определен данный индекс, а также содержит указатель на строки с реальными данными в таблице.

**Типы индексов**

В дополнение к тому, что индекс может быть либо кластеризованным, либо некластеризованным.

**Составной индекс**

Такой индекс может содержать более одного столбца. Вы можете включить до 16 столбцов в индекс, но их общая длина ограничена 900 байтами. Как кластеризованный, так и некластеризованный индексы могут быть составными.

**Уникальный индекс**

Такой индекс обеспечивает уникальность каждого значения в индексируемом столбце. Если индекс составной, то уникальность распространяется на все столбцы индекса, но не на каждый отдельный столбец. Уникальный индекс автоматически создается, когда вы определяете ограничения столбца: первичный ключ или ограничение на уникальность значений:

* *Первичный ключ*  
  Когда вы определяете ограничение первичного ключа на один или несколько столбцов, тогда *SQL Server* автоматически создаёт уникальный кластеризованный индекс, если кластеризованный индекс не был создан ранее (в этом случае создается уникальный некластеризованный индекс по первичному ключу)
* *Уникальность значений*  
  Когда вы определяете ограничение на уникальность значений, тогда *SQL Server* автоматически создает уникальный некластеризованный индекс. Вы можете указать, чтобы был создан уникальный кластеризованный индекс, если кластеризованного индекса до сих пор не было создано на таблице

**Покрывающий индекс**

Такой индекс позволяет конкретному запросу сразу получить все необходимые данные с листьев индекса без дополнительных обращений к записям самой таблицы.

Рекомендации:

* Для таблиц которые часто обновляются используйте как можно меньше индексов.
* Если таблица содержит большое количество данных, но их изменения незначительны, тогда используйте столько индексов, сколько необходимо для улучшение производительности ваших запросов. Однако хорошо подумайте перед использованием индексов на небольших таблицах, т.к. возможно использование поиска по индексу может занять больше времени, нежели простое сканирование всех строк.
* Для кластеризованных индексов старайтесь использовать настолько короткие поля насколько это возможно. Наилучшим образом будет применение кластеризованного индекса на столбцах с уникальными значениями и не позволяющими использовать NULL. Вот почему первичный ключ часто используется как кластеризованный индекс.
* Уникальность значений в столбце влияет на производительность индекса. В общем случае, чем больше у вас дубликатов в столбце, тем хуже работает индекс. С другой стороны, чем больше уникальных значения, тем выше работоспособность индекса. Когда возможно используйте уникальный индекс.
* Для составного индекса возьмите во внимание порядок столбцов в индексе. Столбцы, которые используются в выражениях *WHERE*(к примеру, *WHERE FirstName = 'Charlie'*) должны быть в индексе первыми. Последующие столбцы должны быть перечислены с учетом уникальности их значений (столбцы с самым высоким количеством уникальных значений идут первыми).
* Также можно указать индекс на вычисляемых столбцах, если они соответствуют некоторым требованиям. К примеру, выражение которые используются для получения значения столбца, должны быть детерминистическими (всегда возвращать один и тот же результат для заданного набора входных параметров).

**Запросы к базе данных**

* Старайтесь вставлять или модифицировать в одном запросе как можно больше строк, а не делать это в несколько одиночных запросов.
* Создайте некластеризованный индекс на столбцах которые часто используются в ваших запросах в качестве условий поиска в *WHERE*и соединения в *JOIN*.
* Рассмотрите возможность индексирования столбцов, использующихся в запросах поиска строк на точное соответствие значений.

**Вопросы:**

**Если кластеризованная таблица даёт множество преимуществ, то зачем использовать кучу?**

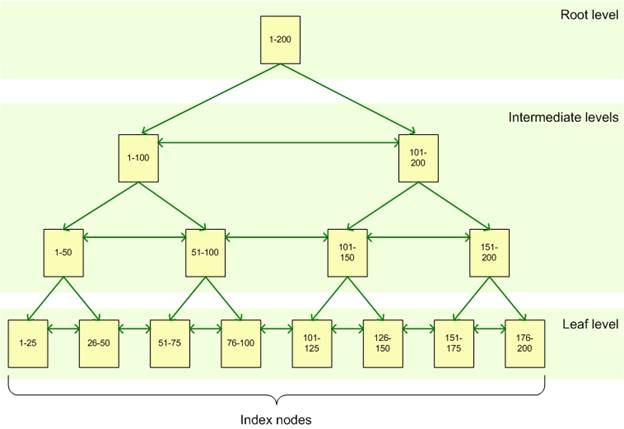
Рассмотрим таблицу с большим количеством вставок, но редкими обновлениями или удалением данных. Отсутствие обновления и удаления данных не должны рассматриваться как единственная причина.  Всё это означает, что вы должны рассматривать возможность использования кучи только когда работаете с особо-маленькими таблицами или всё ваше взаимодействие с таблицей ограничено вставкой данных и ваши запросы чрезвычайно просты (и вы все-равно используете некластеризованные индексы). В противном случае держитесь хорошо спроектированного кластеризованного индекса, к примеру определенного на простом возрастающем ключевом поле, как широко применяемый столбец с *IDENTITY*.\

**Как изменить установленное по умолчанию значение коэффициента заполнения индекса?**

**Какая взаимосвязь между ограничениями на уникальность значения и первичным ключом с индексами таблицы?**

Когда вы создаете первичный ключ, подсистема хранения данных так же создает уникальный кластеризованный индекс, в случае если уже кластеризованный индекс не был создан. Однако, вы можете переопределить установленное по умолчанию поведение и тогда будет создан некластеризованный индекс. Если кластеризованный индекс существует когда вы создаёте первичный ключ, то будет создан уникальный некластеризованный индекс.  
Когда вы создаете ограничение на уникальность, подсистема хранения данных создает уникальный некластеризованный индекс. Но вы можете указать создание уникального кластеризованного индекса, если он не был создан ранее.

**Почему в SQL Server кластеризованные и некластеризованные индексы называются сбалансированным деревом?**

Базовые индексы в SQL Server, кластеризованные или некластеризованные, распространяются по наборам страниц – узлам индекса. Эти страницы организованы в виде определенной иерархии с древовидной структурой, называемой сбалансированным деревом. На верхнем уровне находится корневой узел, на нижнем, конечные узлы листьев, с промежуточными узлами между верхним и нижним уровнями, как показано на рисунке:  
  
Корневой узел предоставляет главную точку входа для запросов, пытающихся получить данные через индекс. Начиная с этого узла, подсистема запросов инициирует переход по иерархической структуре вниз к подходящему конечному узлу, содержащему данные.

**Обязательно ли создавать кластеризованный индекс на столбце с первичным ключом?**

Вы можете создать кластеризованный индекс на любой столбце, соответствующем необходимым условиям. Это верно, что кластеризованный индекс и ограничение первичного ключа созданы друг для друга и их брак заключен на небесах, так что усвойте факт, что когда вы создаете первичный ключ, тогда же будет автоматически создан кластеризованный индекс, если он не был создан ранее.

**А что если проиндексировать представление, то это по-прежнему будет представление?**

Вы можете улучшить производительность запросов, создав кластеризованных индекс и некластеризованные индексы у этого представления, аналогично как вы создаете индексы у таблицы, но основной нюанс состоит в том, что первоначально создается кластеризованный индекс, а затем вы можете создать некластеризованный. Когда вы создаете кластеризованный или некластеризованный индекс у предастваления, то данные физически сохраняются на диск, аналогично обычному индексу. В дополнение, когда в нижележащих таблицах изменяются данные, то индекс представления автоматически изменяется.

**Зачем использовать покрывающий индекс взамен составного индекса?**

**Можно ли создать некластеризованный индекс только для определенного подмножества данных ключевого столбца?**

По умолчанию, некластеризованный индекс содержит по одной строке для каждой строки таблицы. Конечно, вы можете сказать то же самое относительно кластеризованного индекса, принимая в расчет, что такой индекс это и есть таблица. Но что касается некластеризованного индекса, то отношение «один к одному» важный концепт, потому что, начиная с версии *SQL Server 2008*, у вас есть возможность создать фильтруемый индекс, который ограничивает включенные в него строки. Фильтруемый индекс может улучшить производительность выполнения запросов, т.к. он меньше по размеру и содержит отфильтрованную, более аккуратную, статистику, чем вся табличная — это приводит к созданию улучшенных планов выполнения. Фильтруемый индекс также требует меньше места для хранения и меньших затрат на обслуживание. Индекс обновляется только когда изменяются подходящие под фильтр данные.  
В дополнение, фильтруемый индекс легко создать. В операторе *CREATE INDEX* просто необходимо указать в *WHERE*условие фильтрации. К примеру, вы можете отфильтровать из индекса все строки, содержащие NULL, как показано в коде:

Операции создания индексов:

Можно проиндексировать любой столбец таблицы. Это означает, что столбцы, содержащие значения типа данных VARBINARY(max), BIGINT и SQL\_VARIANT, также могут быть индексированы.

Индекс может быть простым или составным. Простой индекс создается по одному столбцу, а составной индекс - по нескольким столбцам. Для составного индекса существуют определенные ограничения, связанные с его размером и количеством столбцов. Индекс может иметь максимум 900 байтов и не более 16 столбцов.

*Параметр UNIQUE* указывает, что проиндексированный столбец может содержать только однозначные (т.е. неповторяющиеся) значения. В однозначном составном индексе однозначной должна быть комбинация значений всех столбцов каждой строки. Если ключевое слово UNIQUE не указывается, то повторяющиеся значения в проиндексированном столбце (столбцах) разрешаются.

*Параметр CLUSTERED* задает кластеризованный индекс, а *параметр NONCLUSTERED* (применяется по умолчанию) указывает, что индекс не изменяет порядок строк в таблице. Компонент Database Engine разрешает для таблицы максимум 249 некластеризованных индексов.

*Параметр INCLUDE* позволяет указать неключевые столбцы, которые добавляются к страницам узлов некластеризованного индекса. Имена столбцов в списке INCLUDE не должны повторяться, и столбец нельзя использовать одновременно как ключевой и неключевой.

**Индексы вычисляемых столбцов**

**Отключение индексов и ограничений**

**Включение индексов и ограничений**

**Изменение индексов**

Компонент Database Engine является одной из немногих систем баз данных, которые поддерживают инструкцию **ALTER INDEX**. Эту инструкцию можно использовать для выполнения операций по обслуживанию индекса. Синтаксис инструкции ALTER INDEX очень сходен с синтаксисом инструкции CREATE INDEX. Иными словами, эта инструкция позволяет изменять значения параметров ALLOW\_ROW\_LOCKS, ALLOW\_PAGE\_LOCKS, IGNORE\_DUP\_KEY и STATISTICS\_NORECOMPUTE, которые были описаны ранее при рассмотрении инструкции CREATE INDEX.

Кроме вышеперечисленных параметров, инструкция ALTER INDEX поддерживает три другие параметра:

* *параметр REBUILD*, используемый для пересоздания индекса;
* *параметр REORGANIZE*, используемый для реорганизации страниц узлов индекса;
* *параметр DISABLE*, используемый для отключения индекса. Эти три параметра рассматриваются в следующих подразделах.

**Пересоздание индекса**

При любом изменении данных, используя инструкции INSERT, UPDATE или DELETE, возможна фрагментация данных. Если эти данные проиндексированы, то также возможна фрагментация индекса, когда информация индекса оказывается разбросанной по разным физическим страницам. В результате фрагментации данных индекса компонент Database Engine может быть вынужден выполнять дополнительные операции чтения данных, что понижает общую производительность системы. В таком случае требуется пересоздать (REBUILD) все фрагментированные индексы.

**Индексы и условия предложения WHERE**

Если условие поиска часто используемого запроса содержит операторы AND, лучше всего будет создать составной индекс по всем столбцам таблицы, указанным в предложении WHERE инструкции SELECT.

**Индексы и оператор соединения**

В случае операции соединения рекомендуется создавать индекс для каждого соединяемого столбца. Соединяемые столбцы часто представляют первичный ключ одной из таблицы и соответствующий внешний ключ другой таблицы. Если указываются ограничения для обеспечения целостности PRIMARY KEY и FOREIGN KEY для соответствующих соединяемых столбцов, следует создать только некластеризованный индекс для столбца внешнего ключа, т.к. система неявно создаст кластеризованный индекс для столбца первичного ключа.

**Покрывающий индекс**