Индексы

Давайте предположим, что у нас есть таблица базы данных, которая имеет более чем 50 миллионов записей с адресами электронной почты, и вы хотите получить одну запись из этой таблицы. Теперь, если вы напишете запрос, чтобы получить адрес электронной почты, MySQL должен будет выполнить проверку в каждой строке в поисках значений, соответствующих вашему запрашиваемому адресу электронной почты. Если MySQL требуется одна микросекунда для сканирования одной записи, то потребуется около пяти секунд, чтобы загрузить всего одну запись, и, поскольку количество записей в таблице увеличивается, затрачиваемое время также будет увеличиваться экспоненциально, что повлияет на производительность

На таблицах могут быть определены индексы, которые быстро извлекают информацию из таблицы без сканирования всей таблицы. Индексы фактически являются указателями на строки в таблице. Определение индекса на таблице не требует каких-либо изменений в фактическом определении таблицы; индексы могут создаваться независимо. Мы можем определить один или несколько индексов или один индекс на нескольких столбцах, исходя из того, как мы будем манипулировать и извлекать данные из таблиц.

Индекс — дополнительная структура, производная от основных данных. Многие БД предоставляют возможность добавлять и удалять индексы без какого-либо воздействия на содержимое базы, это влияет только на производительность запросов.

Общая их идея заключается в дополнительном хранении определенных метаданных, служащих своеобразным дорожным указателем, помогающим найти нужные данные. При необходимости поиска одних и те же данных различными способами может понадобиться несколько разных индексов по различным частям данных.

Поддержка дополнительных структур влечет рост накладных расходов, особенно при записи на диск. Хорошо подобранные индексы ускоряют запросы на чтение, но замедляют запись. Поэтому БД обычно не индексируют по умолчанию все, что можно. При разработке базы данных, нужно выбирать индексы вручную, на основе знания типичных для приложения паттернов запросов. При этом вы можете выбрать те индексы, выгода от которых для приложения максимальна, а накладные расходы не превышают необходимого объема.

Индексирование следует использовать:

- когда требуется группирование на основе определенного столбца;
- когда требуется сортировка по определенному столбцу;
- когда необходимо найти минимальное и максимальное значения в таблице;
- когда имеется большой набор данных и нужно часто отыскивать несколько записей на основе определенных условий;
- когда необходимо выполнять запрос, имеющий соединение между двумя или несколькими таблицами

В MySQL, существует пять типов вариантов индексов:

- первичный (PRIMARY);
- уникальный (UNIQUE);
- столбцовый (COLUMN);
- полнотекстовый (FULLTEXT);
- пространственный (SPATIAL)

Первичные ключи

В любой реляционной таблице должна существовать возможность уникальной идентификации отдельной строки, для этой цели предназначен первичный ключ.

Первичный ключ однозначно идентифицирует одну строку в реляционной таблице, или один документ в документоориентированной базе данных, или одну вершину в графовой. Другие записи в БД могут ссылаться на эту строку/документ/вершину по ее первичному ключу (или идентификатору), а индекс используется для разрешения подобных ссылок.

Для включения столбца в состав первичного ключа используется ключевое слово PRIMARY KEY.

Уникальные ключи (UNIQUE)

Уникальный ключ используется для задания ограничений на столбце, где все значения в данном столбце должны быть неповторяющимися. Уникальные ключи, как и первичные ключи, должны иметь уникальные значения. Однако уникальные ключи допускают значения NULL внутри столбцов. Мы можем определить уникальный ключ на группе столбцов или на одном столбце.

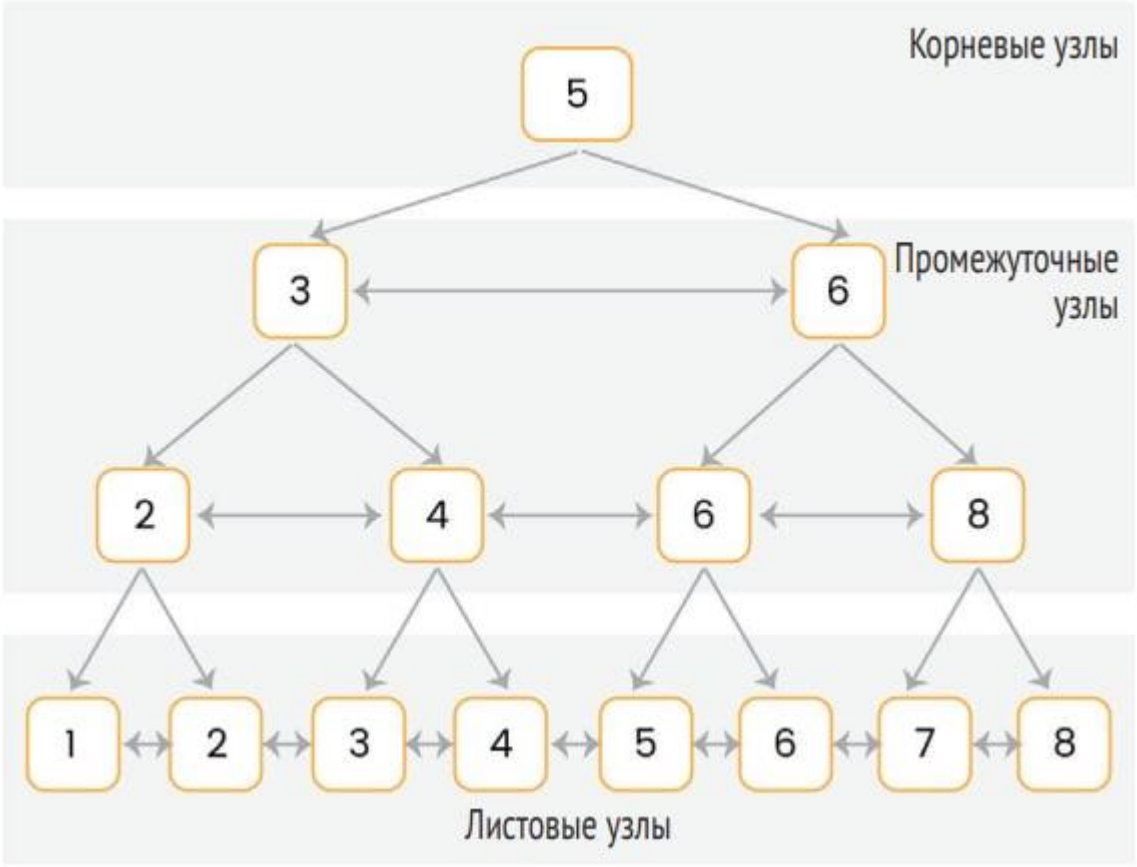
Составные индексы

Очень часто имеется более одного поля, где нам действительно нужно определить индекс для достижения высокой производительности от запроса в крупных наборах данных (в случае запроса нескольких столбцов таблицы
одновременно). Когда индекс создается на нескольких столбцах таблицы, он называется составным индексом. MySQL допускает создание составных индексов максимум на 16 столбцах.
Наиболее распространенный тип просто объединяет несколько полей в один ключ, присоединяя один столбец к другому (в описании индекса указывается, в каком порядке сцепляются поля). Он чем-то похож на старомодную бумажную телефонную книгу с индексом. Благодаря порядку сортировки индекс можно использовать для поиска всех людей с конкретной фамилией или всех людей с конкретным сочетанием «фамилия-имя». Однако он бесполезен, если нужно найти всех людей с конкретным именем.

Ограничения

Ключ	Описание
NOT NULL	Запрет на вставку в столбец неопределенного значения NULL
UNIQUE	Значение столбца должно быть уникальным
PRIMARY KEY	Признак первичного ключа. Значение поля должно быть уникальным, оно не может содержать NULL, в таблице это ограничение может использоваться только один раз
CHECK	Ограничение-проверка на допустимое значение. В скобках за оператором CHECK указывается предикат, проверяющий допустимость значения
FOREIGN KEY и REFERENCES	Оба ограничения весьма похожи, единственное различие между ними в том, что FOREIGN KEY – ограничение внешнего ключа для таблицы, а REFERENCES – ссылка на столбец со значениями подстановки. В случае установки ограничения для таблицы сразу за FOREIGN KEY в скобках необходимо указать перечень столбцов, относящихся к ключу. В остальном синтаксис одинаков. Ограничения внешнего ключа требуют, чтобы все значения, присутствующие во внешнем ключе, соответствовали значениям родительского ключа (обеспечение ссылочной целостности)

BRTREE и hash



BRTREE - индексы на основе B-деревьев

Это структура индексирования, т.е. то, как строится индекс

Ещё один способ построение индекса - использование хеш-таблицы

Хеширование (hashing) полезно в том случае, когда требуется большой объем значений сохранить в сравнительно небольшой по размерности таблице (точнее, хеш-таблице) и обеспечить быстрый доступ к этим значениям

В MySQL 8 хеш-индексы поддерживаются только в подсистемах хранения данных Memory и NDB.