Лекция 13. Индексы и ограничения

В данной лекции мы рассмотрим средства, косвенно влияющие на создаваемый код: индексы и ограничения.

Индексы

Индекс – это механизм поиска определенного элемента ресурса. Например, в конце каждого технического издания есть предметный указатель, позволяющий найти определенное слово или фразу. В указателе эти слова и фразы перечислены в алфавитном порядке, что позволяет читателю быстро перейти к определенной букве, выбрать нужную запись и затем найти страницу или страницы, где можно отыскать это слово или фразу.

Как человек использует предметный указатель, чтобы найти слова в печатном издании, так и сервер БД с помощью индексов выявляет местоположение строк в таблице. ***Индексы*** в БД – это специальные таблицы, содержимое которых, в отличие от обычных таблиц данных, *хранится* в определенном порядке. Однако индекс включает не *все* данные сущности, а только столбец (или столбцы), используемый для определения местоположения строк в таблице данных, а также информацию, описывающую физическое размещение строк. Поэтому предназначение индексов – помочь в извлечении подмножества строк и столбцов таблицы *без* необходимости проверять все строки.

Создание индекса

Рассмотрим таблицу **department**. Допустим, принято решение добавить индекс для столбца *name*, чтобы ускорить выполнение запросов по полному или частичному имени отдела, а также операций *update* или *delete*, использующих это имя. Вот как можно добавить такой индекс в базу данных MySQL:

```
ALTER TABLE department
ADD INDEX dept_name_idx (name);
```

Это выражение создает индекс для столбца *department.name* с именем *dept_name_idx.*

MySQL рассматривает индексы как необязательные компоненты таблицы, вот почему для добавления или удаления индекса используется команда *alter table*. Другие серверы БД, включая SQL Server и Oracle Database, считают индексы независимыми объектами схемы. Поэтому для SQL Server и Oracle индекс формировался бы с помощью команды ***create index***:

```
CREATE INDEX dept_name_idx
ON department (name);
```

Все серверы БД позволяют просматривать доступные индексы. Увидеть все индексы определенной таблицы пользователи MySQL могут с помощью команды ***show***:

```
SHOW INDEX FROM department;
```

Если после создания индекса выясняется, что он не оправдывает себя, его можно удалить следующим образом:

```
ALTER TABLE department
DROP INDEX dept_name_idx;
```

Пользователи SQL Server и Oracle Database для уничтожения индекса должны использовать команду ***drop index***:

```
DROP INDEX dept_name_idx; (Oracle)
DROP INDEX dept_name_idx ON department; (SQL Server)
```

Уникальные индексы

При проектировании БД важно определить, какие столбцы могут содержать дублирующие данные, а какие нет. Правило, запрещающее дублирование строк в определенном столбце, создается путем добавления ***уникального индекса*** *(unique index)*:

```
ALTER TABLE department
ADD UNIQUE dept_name_idx (name);
```

В SQL Server и Oracle Database при создании индекса нужно только добавить ключевое слово *unique*:

```
CREATE UNIQUE INDEX dept_name_idx
ON department (name);
```

Нет необходимости создавать уникальные индексы для столбца(-ов) первичного ключа, поскольку сервер уже проверяет уникальность значений первичных ключей.

Составные индексы

Кроме уже представленных индексов по одному столбцу, можно создавать индексы, охватывающие несколько столбцов, которые называются

составными индексами *(multiple-column indexes)*. Если, например, требуется проводить поиск сотрудников по имени и фамилии, можно сделать индекс сразу для *двух* столбцов:

```
ALTER TABLE employee
ADD INDEX emp_names_idx (lname, fname);
```

Этот индекс будет полезен для запросов, использующих имя и фамилию или только фамилию, но не подходит для запросов, в которых задано только имя сотрудника. Чтобы понять почему, рассмотрим, как проводился бы поиск телефонного номера. Чтобы быстро найти чей-то номер телефона, если известны имя и фамилия, можно воспользоваться телефонной книгой, поскольку она организована по фамилии, а потом по имени. Если известно только имя человека, придется просматривать все записи телефонной книги и выбирать каждую запись с указанным именем.

Поэтому при создании составных индексов необходимо тщательно продумать, какой столбец указывать первым, а какой вторым и т. д., чтобы индекс был максимально полезным.

Ограничения

Ограничение – это просто некоторое ограничивающее условие, налагаемое на один или более столбцов таблицы. Есть несколько разных типов ограничений:

1) *Ограничения первичного ключа (primary-key constraints).* Идентифицируют столбец или столбцы, гарантирующие уникальность в рамках таблицы.

2) *Ограничения внешнего ключа (foreign-key constraints).* На один или более столбцов накладывается такое ограничение: они могут содержать только значения, содержащиеся в столбцах первичного ключа другой таблицы. Также могут ограничиваться допустимые значения других таблиц, если установлены правила *update cascade* (каскадное обновление) или *delete cascade* (каскадное удаление).

3) *Ограничения уникальности (unique constraints).* На один или более столбцов накладывается ограничение: они могут содержать только уникальные в рамках таблицы значения (ограничения первичного ключа – это особый тип ограничений уникальности).

4) *Проверочные ограничения целостности (check constraints).* Ограничивают допустимые значения столбца.

Без ограничений под сомнением может оказаться непротиворечивость базы данных. Например, если сервер допускает изменение ID клиента в таблице **customer** без изменения этого же ID клиента в таблице **account**, все может закончиться тем, что счета больше не будут указывать на соответствующие записи клиентов (это явление известно как «осиротевшие» строки (orphaned rows)). Но если заданы ограничения первичного и внешнего ключей, сервер или сформирует ошибку в случае попытки изменения или удаления данных, на которые ссылаются другие таблицы, или распространит изменения на другие таблицы.

Создание ограничений

Обычно ограничения создают одновременно с ассоциированной таблицей посредством выражения *create table*. Для иллюстрации приведем пример создания таблицы **product**:

```
CREATE TABLE product
 (product_cd VARCHAR(10) NOT NULL,
 name VARCHAR(50) NOT NULL,
 product_type_cd VARCHAR (10) NOT NULL,
 date_offered DATE,
 date_retired DATE,
 CONSTRAINT fk_product_type_cd FOREIGN KEY (product_type_cd)
    REFERENCES product_type (product_type_cd),
 CONSTRAINT pk_product PRIMARY KEY (product_cd)
);
```

Таблица **product** включает два ограничения: первое определяет столбец *product_cd* как первичный ключ таблицы, а второе – столбец *product_type_cd* как внешний ключ к таблице *product_type*. Альтернативный вариант: таблицу **product** можно было создать без ограничений, а ограничения первичного и внешнего ключей добавить позже посредством выражений *alter table*:

```
ALTER TABLE product
ADD CONSTRAINT pk_product PRIMARY KEY (product_cd);

ALTER TABLE product
ADD CONSTRAINT fk_product_type_cd FOREIGN KEY (product_type_cd)
   REFERENCES product_type (product_type_cd);
```

Если требуется убрать ограничения первичного или внешнего ключей, можно снова воспользоваться выражением *alter table*, только в этом случае задается *drop*, а не *add*:

```
ALTER TABLE product
DROP PRIMARY KEY;

ALTER TABLE product
DROP FOREIGN KEY fk_product_type_cd;
```

Ограничение первичного ключа обычно не удаляется, а ограничения внешнего ключа иногда отменяются во время определенных операций обслуживания, а потом устанавливаются вновь.

Каскадные ограничения

Если при наличии ограничений внешнего ключа пользователь пытается вставить новую строку или изменить существующую и при этом получается, что столбец внешнего ключа не имеет соответствующего значения в родительской таблице, сервер формирует ошибку. Однако, можно указать серверу распространять изменение на все дочерние строки, сохраняя, таким образом, целостность данных. Эта разновидность ограничения внешнего ключа известна как ***каскадное обновление*** (*cascading update*) и может быть установлена путем удаления существующего внешнего ключа и добавления нового, включающего блок *on update cascade*:

```
ALTER TABLE product
ADD CONSTRAINT fk_product_type_cd FOREIGN KEY (product_type_cd)
  REFERENCES product_type (product_type_cd)
  ON UPDATE CASCADE;
```

Кроме каскадных обновлений можно задавать ***каскадные удаления*** (*cascading deletes*). При каскадном удалении, если строка удаляется в родительской таблице, соответствующие ей строки удаляются и в дочерней таблице. Для задания каскадного удаления используется блок *on delete cascade*:

```
ALTER TABLE product
ADD CONSTRAINT fk_product_type_cd FOREIGN KEY (product_type_cd)
  REFERENCES product_type (product_type_cd)
  ON UPDATE CASCADE
  ON DELETE CASCADE;
```

Каскадные ограничения – один из случаев, когда ограничения *непосредственно* влияют на код, который вы пишете. Чтобы полностью представлять эффект применения выражений *update* и *delete*, необходимо знать, для каких ограничений базы данных заданы каскадные обновления и/или удаления.

Литература:

1. Алан Бьюли. Изучаем SQL: пер. с англ. – СПб-М.: Символ, O'Reilly, 2007. – 310 с.