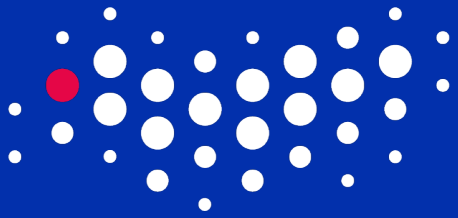

УНИВЕРСИТЕТ ИТМО

Объекты БД

Чертков Виталий

Санкт-Петербург, 2017

Объекты БД

- Индекс
- Представление
- Триггер

INDEX

Индекс (англ. index) — объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы , таким образом, позволяет искать строки, удовлетворяющие критерию поиска. Ускорение работы с использованием индексов достигается в первую очередь за счёт того, что индекс имеет структуру, оптимизированную под поиск — например, сбалансированного дерева.

Как создать индекс ?

Шаблон:

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name
        ON tbl_name (col_name[(length)],... )

CREATE INDEX ind_user_login
        ON user (login);
```

Созданный индекс должен использоваться при выполнении запроса.

```
EXPLAIN SELECT * FROM user
        WHERE login LIKE “Zero”;
```

Удаление индекса.

Шаблон:

```
DROP INDEX index_name ON tbl_name
```

Пример:

```
DROP INDEX ind_user_login ON user;
DROP INDEX `PRIMARY` ON user;
```

VIEW

Представление (view) — виртуальная (логическая) таблица, представляющая собой результат выполнения хранимого запроса (SELECT) в БД.

В отличие от обычных таблиц, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице базы данных немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы.

Применение представления дает:

1. Дает возможность гибкой настройки прав доступа к данным за счет того, что права даются не на таблицу, а на представление.

2. Позволяет разделить логику хранения данных и программного обеспечения. Можно менять структуру данных, не затрагивая программный код.

3.Удобство в использовании за счет автоматического выполнения таких действий как доступ к определенной части строк и/или столбцов индекса.

Шаблон создание представления:

```
CREATE  VIEW view_name [(column_list)]
    AS select_statement
```

Пример:

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price,
qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

Какие операции разрешены над представлением?

UPDATE, DELETE, [INSERT]

TRIGGER

Это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено наступлением определенного события (действием) — по сути добавлением INSERT или удалением DELETE строки в заданной таблице, или модификации UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Зачем использовать триггеры ?

Для автоматической генерации значений виртуального поля

Для логгирования

Для сбора статистики

Для изменения данных в таблицах, если в dml операции участвует представление

Для предотвращения dml операций в какие-то определенные часы

Для реализации сложных ограничений целостности данных, которые невозможно осуществить через описательные ограничения, установленные при создании таблиц

Для организации всевозможных видов аудита

Для оповещения других модулей о том, что делать в случае изменения информации в БД

Для реализации бизнес логики

Для организации каскадных воздействий на таблицы БД

Для отклика на системные события в БД или схеме

Шаблон создания триггера:

```
CREATE TRIGGER trigger_name
    ON tbl_name  trigger_time trigger_event
   FOR EACH ROWBEGIN
   BEGIN
     trigger_body
   END;

trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
```

Пример создания журнала.

```
CREATE TABLE test (
 id INT(11) NOT NULL AUTO_INCREMENTPRIMARY KEY,
 content TEXT NOT NULL)

CREATE TABLE backup (
 id INT(11)NOT NULL AUTO_INCREMENT PRIMARY KEY,
 row_id INT(11) UNSIGNED NOT NULL,
 content TEXT NOT NULL )

DELIMITER $$
CREATE TRIGGER update_test before update ON test
FORE ACH RAW
 BEGIN
  INSERT INTO backup set row_id = OLD.id,
content = OLD.content;
 END;
```

Строковый триггер срабатывает один раз для каждой строки. При этом внутри триггера можно обращаться к строке обрабатываемой в данный момент времени. Делать это можно, применяя псевдозаписи: **old.column_name**
и **new.column_name**

Значение псевдозаписи при различных операциях:

Активизирующий оператор	OLD.	NEW.
INSERT	Не определена во всех полях содержится NULL значения	Значения, которые будут введены после выполнения оператора.
UPDATE	Исходные значения содержащиеся в строке перед обновлением данных	Новые значения которые будут введены после выполнения оператора
DELETE	Исходные значения содержащиеся в строке перед ее удалением	Не определена во всех полях содержится NULL значения

Триггер с использованием переменных:

```
DELIMITER $$

CREATE TRIGGER `update_test` before
update ON `test`
FOR EACH ROW
BEGIN
   set @row_id = OLD.id;
   set @content = OLD.content;
   INSERT INTO backup(row_id, content,
dt_create) values(@row_id, @content,
now());
END;
```

УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!

Санкт-Петербург, 2017