ЛР4. Представления. Триггеры

Представления

https://dev.mysql.com/doc/refman/8.0/en/views.html

Представление (`VIEW`) - объект базы данных, представляющий собой виртуальную таблицу. В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Преимущества использования представлений:

1. Дает возможность гибкой настройки прав доступа к данным за счет того, что права даются не на таблицу, а на представление. Это позволяет назначить права на отдельные строки таблицы или возможность получения не самих данных, а результата каких-то действий над ними.
2. Позволяет разделить логику хранения данных и программного обеспечения. Можно менять структуру данных, не затрагивая программный код. При изменении схемы БД достаточно создать представления, аналогичные таблицам, к которым раньше обращались приложения. Это очень удобно, когда нет возможности изменить программный код или к одной базе данных обращаются несколько приложений с различными требованиями к структуре данных.
3. Удобство в использовании за счет автоматического выполнения таких действий как доступ к определенной части строк и/или столбцов, получение данных из нескольких таблиц и их преобразование с помощью различных функций.
4. Поскольку SQL-запрос, выбирающий данные представления, зафиксирован на момент его создания, СУБД получает возможность применить к этому запросу оптимизацию или предварительную компиляцию, что может повысить производительность выполнения запросов.

Создание представления (упрощенный синтаксис):

```
CREATE [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Данное выражение создает новое представление или заменяет существующее, если используется выражение `OR REPLACE`. Выражение `select_statement` обеспечивает определение представления. SELECT может относиться как к таблицам, так и к другим представлениям, т.е. могут быть вложенными.

Представления должны иметь уникальные имена столбцов. По умолчанию имена столбцов, полученные выражением `SELECT`, используются для имени столбца представления. Явные имена для столбцов представления могут быть заданы с помощью выражения `column_list` как список разделяемых запятой идентификаторов.

Опция `ALGORITHM` определяет, как MySQL обрабатывает представления:

- `MERGE`: текст запроса к представлению объединяется с текстом запроса самого представления к таблицам БД, результат выполнения объединенного запроса возвращается пользователю.
- `TEMPTABLE`: содержимое представлению сохраняется во временную таблицу, которая затем используется для выполнения запроса.
- `UNDEFINED` (значение по умолчанию): MySQL самостоятельно выбирает какой алгоритм использовать при обращении к представлению.

Представление называется *обновляемым*, если к нему могут быть применимы операторы `UPDATE` и `DELETE` для изменения данных в таблицах, на которых основано представление. Для того чтобы представление было обновляемым, должны быть выполнены некоторые условия, в т.ч.:

1. Отсутствие функций агрегации в представлении.
2. Отсутствие следующих выражений в представлении: DISTINCT, GROUP BY, HAVING, UNION.
3. Отсутствие подзапросов в списке выражения SELECT
4. Колонки представления быть простыми ссылками на поля таблиц (а не, например, арифметическими выражениями) и т.д.

Обновляемое представление может допускать добавление данных (`INSERT`), если все поля таблицы-источника, не присутствующие в представлении, имеют значения по умолчанию. Для представлений, основанных на нескольких таблицах, операция добавления данных (`INSERT`) работает только в случае если происходит добавление в единственную реальную таблицу. Удаление данных (`DELETE`) для таких представлений не поддерживается.

При использовании в определении представления конструкции `WITH [CASCADED | LOCAL] CHECK OPTION` все добавляемые или изменяемые строки будут проверяться на соответствие определению представления.

- Изменение данных (`UPDATE`) разрешено только для строк, удовлетворяющих условию `WHERE` в определении представления. Кроме того, новые значения строки также должны удовлетворять значениями удовлетворяет условию `WHERE`.
- Добавление данных (`INSERT`) будет происходить, только если новая строка удовлетворяет условию `WHERE` в определении представления.

Ключевые слова `CASCADED` и `LOCAL` определяют глубину проверки для представлений, основанных на других представлениях:

- Для `LOCAL` происходит проверка условия `WHERE` только в собственном определении представления.
- Для `CASCADED` происходит проверка для всех представлений на которых основано данное представление. Значением по умолчанию является `CASCADED`.

В начале выполнения работы следует выполнить запрос

```
SET SQL_SAFE_UPDATES = 0
```

При значении параметра `SQL_SAFE_UPDATES = 1` выполнение команды `UPDATE` возможно только в том случае, если в запросе указан первичный ключ.

Триггеры

https://dev.mysql.com/doc/refman/8.0/en/triggers.html

Триггер представляет собой именованный объект базы данных, который связан с таблицей. Он будет активирован, когда произойдет определенное событие (`INSERT, DELETE, UPDATE`), связанное с заданной таблицей. Триггер может быть установлен для активации до или после события. Триггеры не могут быть использованы с представлениями.

Синтаксис создания триггера:

```
CREATE
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

Данное выражение создает новый триггер. *trigger_name* – название триггера. *trigger_time* определяет время выполнения триггера: до или после наступления события *trigger_event*. В качестве события могут быть следующие: `INSERT, UPDATE, DELETE`.

Возможно создание нескольких триггеров для заданной таблицы, с одинаковыми значениями *trigger_time* и *trigger_event*. В данном случае по умолчанию порядок выполнения таких триггеров определяется датой их создания. Можно изменить порядок выполнения таких триггеров, используя выражения `FOLLOWS` и `PRECEDES`. С `FOLLOWS` новый триггер активируется после существующего триггера. С `PRECEDES` новый триггер активируется перед существующим триггером.

`trigger_body` представляет выражение, которое выполняется при активации триггера. Если выражение состоит из нескольких строк (операций), разделенных «;», то данное выражение заключается внутри конструкции BEGIN…END. Подробное описание синтаксиса операций, заключенных внутри данной конструкции,

представлено здесь: https://dev.mysql.com/doc/refman/8.0/en/sql-compound-statements.html.

В теле триггера ключевые слова `OLD` и `NEW` позволяют получить доступ к строкам таблицы, действие над которыми активировали триггер. В случае `INSERT` триггера может использоваться только `NEW.col_name`, так как нет старых строк. В `DELETE` триггере может использоваться только `OLD.col_name`, так как нет новых строк. В `UPDATE` триггере можно использовать `OLD.col_name` для обращения к столбцам строки перед ее обновлением и `NEW.col_name` для обращения к столбцам строки после ее обновления.

```
CREATE TRIGGER upd_check BEFORE UPDATE ON account
       FOR EACH ROW
       BEGIN
           IF NEW.amount < 0 THEN
               SET NEW.amount = 0;
           ELSEIF NEW.amount > 100 THEN
               SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'error';
           END IF;
       END;//
```

При написании кода триггеров следует использовать оператор DELIMITER, который заменяет стандартный разделитель «;» на указанный пользователем. Это нужно для того, чтобы сервер MySQL воспринимал объявление функции, процедуры или триггера как единый запрос.

Пример:

```
delimiter //
    CREATE TRIGGER upd_check BEFORE UPDATE ON account
    FOR EACH ROW
    BEGIN
      …
    END;//
delimiter ;
```