ЛР5. Хранимые процедуры и функции. Курсоры

Хранимые процедуры и функции

https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры могут иметь входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных. Кроме того, в хранимых процедурах возможны циклы и ветвления.

Упрощенный синтаксис создания хранимой процедуры:

```
CREATE
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

type:
    Any valid MySQL data type

characteristic:
    COMMENT 'string'
  | [NOT] DETERMINISTIC
routine_body:
    Valid SQL routine statement
```

`sp_name` – название хранимой процедуры. Список параметров `proc_parameter`, заключенный в круглые скобки, должен присутствовать всегда. Если параметров нет, следует использовать пустой список параметров (). Имена параметров не чувствительны к регистру.

Каждый параметр является параметром `IN` по умолчанию. Чтобы указать иное для параметра, используется ключевое слово `OUT` или `INOUT` перед именем параметра. Параметр `IN` передает значение в процедуру. Процедура может изменить значение, но изменение не будет видно вызывающей стороне, когда процедура выполнится. Параметр `OUT` передает значение из процедуры обратно вызывающей стороне. Его начальное значение равно `NULL` в процедуре, и его значение видно вызывающей

стороне, когда процедура выполнится. Параметр INOUT инициализируется вызывающей стороной, может быть изменен процедурой, и любое изменение, внесенное процедурой, будет видно вызывающей стороне, когда процедура выполнится.

Характеристика COMMENT может использоваться для описания хранимой процедуры. Эта информация отображается в инструкции SHOW CREATE PROCEDURE.

Процедура считается «детерминированной», если она всегда дает один и тот же результат для одних и тех же входных параметров, и «недетерминированной» в противном случае. Если ни DETERMINISTIC, ни NOT DETERMINISTIC не указаны в определении процедуры, по умолчанию используется значение NOT DETERMINISTIC. MySQL не проверяет, что процедура, объявленная DETERMINISTIC, не содержит операторов, которые дают недетерминированные результаты, и наоборот. Однако неправильное указание типа процедуры может повлиять на результаты или производительность.

Тело процедуры routine_body состоит из выражений языка SQL. Это может быть простой оператор, такой как SELECT или INSERT, или составной оператор, написанный с использованием BEGIN и END. Составные операторы могут содержать объявления, циклы и другие операторы, структуры управления. Синтаксис этих операторов подробно описан здесь: https://dev.mysql.com/doc/refman/8.0/en/sql-compound-statements.html.

Для вызова процедур используется оператор CALL.

Синтаксис создания функций имеет схожий вид:

```
CREATE
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body
```

Указание параметра как IN, OUT или INOUT действительно только для процедур. Для функций все параметры всегда рассматриваются как параметры IN. Выражение RETURNS указывает тип возвращаемого значения функции, а тело функции должно содержать оператор RETURN. Если инструкция RETURN возвращает значение другого типа, оно приводится к нужному типу.

В отличие от хранимых процедур, функции могут быть вызваны внутри SQL выражения (например, `SELECT func_sum(x, y) FROM numbers`).

Курсоры

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

Курсор представляет собой указатель на одну строку из набора строк. Курсор используются, когда требуется осуществлять обработку данных построчно. При использовании курсора в MySQL нельзя обновить данные строк, обход строк производится только в одном направлении, так же нельзя пропускать строки.

Курсор объявляется с помощью выражения `DECLARE`:

```
DECLARE cursor_name CURSOR FOR SELECT_statement
```

Здесь `cursor_name` – название курсора, `SELECT_statement` – выражение, описывающее выборку данных, по которой буде произведен обход. Курсор должен быть объявлен после объявления всех переменных.

Для открытия курсора используется выражение `OPEN`:

```
OPEN cursor_name
```

Оператор `OPEN` инициализирует результирующий набор для курсора, поэтому он должен быть вызван до начала обхода строк из результирующего набора.

Затем используется оператор `FETCH`, чтобы получить строку, указанную курсором, и переместить курсор на следующую строку результирующего набора:

```
FETCH cursor_name INTO variables_list
```

Здесь `variables_list` – список переменных, в которые извлекаются данные, описанные в `SELECT_statement`.

После этого выполняется проверка, доступны ли еще строки для извлечения. При работе с курсором необходимо определить, что делать в случае, когда курсор не может найти следующую строку. Для этого следует объявить обработчик события `NOT FOUND` следующим образом:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1
```

Здесь `finished` – переменная, которая сигнализирует о достижении конца результирующего набора. Обработчик событий должен объявляться после объявления курсора.

В конце используется выражение `CLOSE` для освобождения памяти, используемой курсором:

```
CLOSE cursor_name
```

Данная диаграмма иллюстрирует работу курсора: