数据库技术与应用

第四章 MYSQL触发器与存储过程

本章主要内容

- 4.1 存储过程和函数
  - 4.1.1 创建与修改存储过程或函数
  - 4.1.2 删除存储过程或函数
  - 4.1.3 查看存储过程或者函数
  - 4.1.4 使用变量
  - 4.1.5 定义条件和处理
  - 4.1.6 光标的使用
  - 4.1.7 流程控制语句
- 4.2 触发器
  - 4.2.1 创建触发器
  - 4.2.2 删除与查看触发器
  - 4.2.3 触发器实现CHECK、级联

4.1 存储过程和函数

存储程序和函数是事先经过编译并存储在数据库中的一套SQL语句。创建后，客户端不需要再重新发布单独的语句，而是可以引用存储程序来替。存储程序可以提供改良后的性能，因为只有较少的信息需要在服务器和客户算之间传送，代价是增加数据库服务器系统的负荷。存储程序也允许你在数据库服务器上有函数库，从而提高了开发人员的开发效率。

v 以下情况存储程序尤其有用：

* 当用不同语言编写多客户应用程序，或多客户应用程序在不同平台上运行且需要执行相同的数据库操作之时。
* 安全极为重要之时。比如，银行对所有普通操作使用存储程序。程序可以确保每一个操作都被妥善记入日志。应用程序和用户不可能直接访问数据库表，但是仅可以执行指定的存储程序。

4.1.1 创建与修改存储过程或函数

v存储过程示例1—返回单个数据

```
mysql> delimiter $$
mysql> CREATE PROCEDURE qq_count (OUT num INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO num FROM qq;
    -> END;
    -> $$
mysql> delimiter ;
mysql> CALL qq_count(@a);
mysql> SELECT @a;
```

```
mysql> select @a;
+------+
| @a   |
+------+
|    6 |
+------+
1 row in set (0.00 sec)
```

v存储过程示例2—返回数据集

```
CREATE PROCEDURE getrecord()
BEGIN
   SELECT * FROM qq;
END;
```

```
mysql> call getrecord();
+-----------+-----+-------------------+
| qqno      | tid | email             |
+-----------+-----+-------------------+
| 456789797 |   1 | zzhstudio@126.com |
| 7979879   |   2 | zzh@126.com       |
| 657897897 |   3 | zzhstudio@163.com |
| 7979879   |   4 | 7979879@qq.com    |
| 56498779  |   5 | zzh@163.com       |
| 55877455  |   6 | 55877455@qq.com   |
+-----------+-----+-------------------+
6 rows in set (0.00 sec)
```

4.1.1 创建与修改存储过程或函数

v存储函数的创建示例

```
mysql> delimiter $$
mysql> CREATE FUNCTION id_email(id int) RETURNS varchar(20)
       Reads SQL data
    -> BEGIN
    ->   declare email varchar(20);
    ->   SELECT email INTO email FROM qq where tid=id;
    ->   RETURN email;
    -> END;
    -> $$
mysql> delimiter ;
mysql> CALL id_email(1);
mysql> SELECT id_email(1);
```

```
mysql> select * from qq;
+-----------+-----+-------------------+
| qqno      | tid | email             |
+-----------+-----+-------------------+
| 456789797 |   1 | zzhstudio@126.com |
| 7979879   |   2 | zzh@126.com       |
| 657897897 |   3 | zzhstudio@163.com |
| 7979879   |   4 | 7979879@qq.com    |
| 56498779  |   5 | zzh@163.com       |
| 55877455  |   6 | 55877455@qq.com   |
+-----------+-----+-------------------+
```

```
mysql> select id_email(1);
+-------------------+
| id_email(1)       |
+-------------------+
| zzhstudio@126.com |
+-------------------+
1 row in set (0.00 sec)
```

说明：函数的创建需要指定返回值类型；同时应当在定义体中指明返回的结果。调用函数不能使用CALL语句，函数可直接使用。

存储过程或函数基本语法

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

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

    proc_parameter:    [ IN | OUT | INOUT ] param_name type
    func_parameter:    param_name type(Any valid MySQL data type)

    characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

routine_body:
    Valid SQL procedure statement or statements
```

存储过程或函数说明

- 存储子程序的权限问题。
  - 创建存储子程序需要CREATE ROUTINE权限。
  - 修改或移除存储子程序需要ALTER ROUTINE权限。这个权限自动授予子程序的创建者。
  - 执行子程序需要EXECUTE权限。然而，这个权限自动授予子程序的创建者。
- RETURNS字句只能对FUNCTION做指定，对函数而言这是强制的。它用来指定函数的返回类型，而且函数体必须包含一个RETURN value语句。

存储过程或函数说明

- 子程序与当前数据库关联。要明确地把子程序与给定数据库关联起来，在创建子程序时指定其名字为db_name.sp_name。
  - 当一个子程序被调用时，一个隐含的USE db_name 被执行（当子程序终止时停止执行）。存储子程序内的USE语句时不允许的。
  - 你可以使用数据库名限定子程序名。这可以被用来引用一个不在当前数据库中的子程序。比如，要引用一个与test数据库关联的存储程序p或函数f，你可以说CALL test.p()或test.f()。
  - 数据库移除的时候，与它关联的所有存储子程序也都被移除。

存储过程或函数说明

v 参数说明：

* 由括号包围的参数列必须总是存在。如果没有参数，也该使用一个空参数列()。
* 每个参数默认都是一个IN参数。要指定为其它参数，可在参数名之前使用关键词 OUT或INOUT
* 指定参数为IN, OUT, 或INOUT 只对PROCEDURE是合法的。FUNCTION参数总是被认为是IN参数

存储过程或函数说明

v Characteristic特性说明：

* CONTAINS SQL表示子程序不包含读或写数据的语句。NO SQL表示子程序不包含SQL语句。READS SQL DATA表示子程序包含读数据的语句，但不包含写数据的语句。MODIFIES SQL DATA表示子程序包含写数据的语句。如果这些特征没有明确给定，默认的是CONTAINS SQL。

* SQL SECURITY特征可以用来指定子程序该用创建子程序者的许可来执行，还是使用调用者的许可来执行。默认值是DEFINER。

* COMMENT子句是一个MySQL的扩展，它可以被用来描述存储程序。这个信息被SHOW CREATE ROCEDURE和 SHOW CREATE FUNCTION语句来显示。

存储过程或函数说明

v 过程体说明：

* MySQL允许子程序包含DDL语句，如CREATE和DROP。MySQL也允许存储程序（但不是存储函数）包含SQL 交互select语句。
* 存储函数不可以包含那些做明确的和绝对的提交或者做回滚的语。
* 存储子程序不能使用LOAD DATA INFILE。
* 返回结果包的语句不能被用在存储函数中。
* 其它语句：块语句、选择、循环等等

存储过程或函数说明

- BEGIN ... END复合语句块：

[begin_label:] BEGIN

[statement_list]

END [end_label]

* 存储子程序可以使用BEGIN ... END来包含多个语句。
* statement_list代表一个或多个语句的列表，每个语句都必须用分号（；）来结尾。
* 复合语句可以被标记。除非begin_label存在,否则end_label不能被给出,并且如果二者都存在,他们必须是同样的。

4.1.2 修改和删除存储过程或函数

v修改语法

```
ALTER {PROCEDURE | FUNCTION} sp_name [characteristic ...]
characteristic:
    { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES
    SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

说明：如果要重新完整的定义已有的存储过程，建议采用先删除该存储过程后，然后再进行创建。

v删除语法

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

IF EXISTS子句是一个MySQL的扩展。如果程序或函数不存储，它防止发生错误。

4.1.3 查看存储过程或者函数

- SHOW CREATE {PROCEDURE | FUNCTION} sp_name

  SHOW CREATE PROCEDURE getrecord;

- SHOW {PROCEDURE | FUNCTION} STATUS [LIKE 'pattern']

  SHOW PROCEDURE status like ‘getrecord’;

- 查看系统表information_schema.Routines

  SELECT * FORM Routines where routine_name=‘getrecord’;

4.1.4 使用变量

v 使用变量的基本步骤

* 1 DECLARE局部变量

DECLARE *var_name*[,...] *type* [DEFAULT *value*]

- DECLARE仅被用在BEGIN ... END复合语句里，并且必须在复合语句的开头，在任何其它语句之前。
- DEFAULT子句指定默认值（常量或表达式），不指定则初始值为NULL。

* 2 SET语句为变量赋值

SET var_name = expr [, var_name = expr] ...

- SET语句可以同时给多个变量赋值。

4.1.4 使用变量

* SELECT ... INTO语句

SELECT *col_name*[,...] INTO *var_name*[,...] *table_expr*

- 把选定的多个字段直接存储到变量。
- 只有一条记录的字段可以被取回。

```
DELIMITER $$
CREATE  PROCEDURE  qq_t_var()
begin
  declare id int;
  declare name varchar(10);
  declare mail varchar(25);
  set id=1;
  select tid,qqname,email into id,name,mail from qq_t where tid=id;
  select id,name,mail;
end $$
DELIMITER ;
```

```
mysql> select * from qq_t;
+-----------+-----+--------+-------------------+
| qqno      | tid | qqname | email             |
+-----------+-----+--------+-------------------+
| 23432435  |   1 | 天使   | zzhstudio@126.com |
+-----------+-----+--------+-------------------+
1 row in set (0.00 sec)
```

```
mysql> call qq_t_var();
+------+------+-------------------+
| id   | name | mail              |
+------+------+-------------------+
|    1 | 天使 | zzhstudio@126.com |
+------+------+-------------------+
1 row in set (0.00 sec)
```

4.1.5 定义条件和处理

定义条件和处理程序能够事先定义程序执行过程中有可能遇到的问题，并采用一定的机制解决相关问题。

v 基本步骤

* 1 DECLARE条件的定义

DECLARE *condition_name* CONDITION FOR *condition_value*

- **condition_name**：条件的名称
- **condition_value**：可以取sqlstate_value或mysql_error_code，都表示MYSQL的错误代码。

  例如：下面插入重复主键值的错误代码为ERROR 1062(23000)，则sqlstate_value=23000， mysql_error_code=1062

```
mysql> insert into qq(qqno,tid) values('68747539',1),('705646790',1);
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

4.1.5 定义条件和处理

2 条件的处理

DECLARE handler_type HANDLER FOR condition_value[,...] sp_statement

handler_type: CONTINUE | EXIT | UNDO(未支持)

condition_value: sqlstate_value | condition_name | SQLWARNING | NOT FOUND | SQLEXCEPTION | mysql_error_code

- SQLWARNING是对所有以01开头的SQLSTATE代码的速记。
- NOT FOUND是对所有以02开头的SQLSTATE代码的速记。
- SQLEXCEPTION是对所有没有被SQLWARNING或NOT FOUND捕获的SQLSTATE代码的速记。

4.1.5 定义条件和处理—示例

```
mysql> delimiter$$
mysql> CREATE PROCEDURE qq_condition ()
    -> BEGIN
    ->   DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
    ->   SET @x = 1;
    ->   INSERT INTO qq(qqno,tid) VALUES ('68747539',1);
    ->   SET @x = 2;
    ->   INSERT INTO qq(qqno,tid) VALUES ('705646790',1);
    ->   SET @x = 3;
    -> END;
    -> $$
```

```
mysql> insert into qq(qqno,tid) values('68747539',1),('705646790',1);
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

```
mysql> call qq_condition();
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select @x,@x2;
+------+------+
| @x   | @x2  |
+------+------+
|    3 |    1 |
+------+------+
1 row in set (0.00 sec)
```

将上面存储过程句柄改为exit。

```
DECLARE EXIT HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
```

```
mysql> call qq_condition_exit();
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select @x,@x2;
+------+------+
| @x   | @x2  |
+------+------+
|    2 |    1 |
+------+------+
1 row in set (0.00 sec)
```

4.1.5 定义条件和处理

v 其他条件形式

```
mysql> insert into qq(qqno,tid) values('68747539',1),('705646790',1);
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

* 捕获mysql_error_code

  DECLARE CONTINUE HANDLER FOR 1062 SET @x2 = 1;

* 事先定义condition_name

  DECLARE DuplicateKey CONDITION FOR SQLSTATE '23000';

  DECLARE CONTINUE HANDLER FOR DuplicateKey SET @x2 = 1;

* 捕获SQLEXCEPTION

  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET @x2 = 1;

4.1.6 光标的使用

利用基于变量的select into语句，仅能处理一条记录的数据。通过光标（或游标），能够对查询的结果集进行循环处理。

v 基本步骤

*1. 声明光标： DECLARE cursor_name CURSOR FOR select_statement

*2. 打开光标： OPEN cursor_name

*3. 使用光标： FETCH cursor_name INTO var_name [, var_name] ...

*4. 关闭光标： CLOSE cursor_name

说明：

1、光标必须在声明处理程序之前被声明，并且声明变量和条件之后。

2、SELECT语句中不能有INTO子句。

4.1.6 光标的使用—示例

利用光标，创建一存储过程实现将qq_t表中的数据添加到qq表中。

```
CREATE PROCEDURE qq_cursor_insert ()
BEGIN
  DECLARE a varchar(15);
  DECLARE b int;
  DECLARE c varchar(10);
  DECLARE d varchar(25);
  DECLARE cur1 CURSOR FOR SELECT qqno,tid,qqr
  DECLARE exit HANDLER FOR NOT FOUND CLOS
  OPEN cur1;
  REPEAT
      FETCH cur1 INTO a,b,c,d;
      insert into qq(qqno,tid,qqname,email) values(a,b,c,
   UNTIL 0 END REPEAT;
   CLOSE cur1;
END
```

```
mysql> select * from qq;
Empty set (0.00 sec)

mysql> select * from qq_t;
+----------+-----+--------+-------------------+
| qqno     | tid | qqname | email             |
+----------+-----+--------+-------------------+
| 23432435 |   1 | 天使   | zzhstudio@126.com |
| 9898080  |   2 | 钟老师 | zzh@126.com       |
+----------+-----+--------+-------------------+
2 rows in set (0.06 sec)

mysql> call qq_cursor_insert ();
Query OK, 0 rows affected, 1 warning (0.05 sec)

mysql> select * from qq;
+----------+-----+--------+-------------------+
| qqno     | tid | qqname | email             |
+----------+-----+--------+-------------------+
| 23432435 |   1 | 天使   | zzhstudio@126.com |
| 9898080  |   2 | 钟老师 | zzh@126.com       |
+----------+-----+--------+-------------------+
2 rows in set (0.00 sec)
```

4.1.7 流程控制语句

- IF语句
- CASE语句
- LOOP语句
- LEAVE语句
- ITERATE语句
- REPEAT语句
- WHILE语句

参看教材P316

4.2 触发器

触发器（Trigger）是用户定义在关系表上的一类由事件驱动(insert、update、delete)的特殊过程。利用触发器能够有效地保证数据完整性，也便于执行一些自动的数据操作。

- v4.2.1 创建触发器
- v4.2.2 触发器实现级联、CHECK
- v4.2.3 删除与查看触发器

4.2.1 创建触发器

v触发器创建的基本语法：

CREATE TRIGGER 触发器名 BEFORE|AFTER 触发事件 ON 表名 FOR EACH ROW 触发器语句

* 触发器只能建立在基本表上，不能建立在临时表或视图上。
* 触发事件包括：INSERT、UPDATE、DELETE
* FOR EACH ROW行级触发器，MYSQL目前还没有支持语句级的触发器。
* 一个表上不能有两个相同时间和事件的触发程序。
* MYSQL中，触发器执行的顺序是BEFORE触发器、表操作(INSERT、UPDATE 和 DELETE)、AFTER触发器。

4.2.1 创建触发器

* 使用OLD(旧值)和NEW(新值)关键字，能够访问受触发程序影响的行中的字段值（OLD和NEW不区分大小写）。
* 在INSERT触发程序中，仅能使用NEW.col_name，没有旧值。
* 在DELETE触发程序中，仅能使用OLD.col_name，没有新值。
* 在UPDATE触发程序中，可以使用OLD.col_name来引用更新前的旧值，也能使用NEW.col_name来引用更新后的行中的新值。
* OLD列是只读的，可以引用它，但不能更改它。
* NEW列，如果具有SELECT权限，可引用它。在BEFORE触发程序中，如果具有UPDATE权限，可使用“SET NEW.col_name = value”更改它的值。意味着可以使用触发程序来更改将要插入到新行中的值，或用于更新行的值。

4.2.2 触发器实现级联

利用触发器实现学生与成绩表学生编号上的级联更新。

```
create trigger stud_update after u
begin
  declare newsno char(10);
  declare oldsno char(10);
  set newsno=new.sno;
  set oldsno=old.sno;
  if newsno<>oldsno then
     update scsc set sno=newsno wh
  End if;
END;
```

```
mysql> select * from stud;
+------------+-------+------+-----+--------+
| sno        | sname | ssex | sage | sdep   |
+------------+-------+------+-----+--------+
| 2009010101 | 张三  | M    |   20 | 计科系 |
| 2009010102 | 李四  | M    |   21 | 计科系 |
+------------+-------+------+-----+--------+
2 rows in set (0.00 sec)

mysql> select * from scsc;
+------------+------+-------+
| sno        | cno  | grade |
+------------+------+-------+
| 2009010101 | 1001 |    70 |
| 2009010102 | 1002 |    85 |
+------------+------+-------+
2 rows in set (0.00 sec)

mysql> update stud set sno='2009010103' where sname='张三';
Query OK, 1 row affected (0.16 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from scsc;
+------------+------+-------+
| sno        | cno  | grade |
+------------+------+-------+
| 2009010103 | 1001 |    70 |
| 2009010102 | 1002 |    85 |
+------------+------+-------+
2 rows in set (0.00 sec)
```

4.2.2 触发器实现级联

利用触发器实现学生与成绩表学生编号上的级联删除。

```
create trigger stud_delete after delete on stud for each row
begin
  declare oldsno char(10);
  set oldsno=old.sno;
  delete from scsc where sno=oldsno;
END;
```

```
mysql> select * from scsc;
+------------+------+-------+
| sno        | cno  | grade |
+------------+------+-------+
| 2009010103 | 1001 |    70 |
| 2009010102 | 1002 |    85 |
+------------+------+-------+
2 rows in set (0.00 sec)

mysql> select * from stud;
+------------+-------+------+------+--------+
| sno        | sname | ssex | sage | sdep   |
+------------+-------+------+------+--------+
| 2009010102 | 李四  | M    |   21 | 计科系 |
| 2009010103 | 张三  | M    |   20 | 计科系 |
+------------+-------+------+------+--------+
2 rows in set (0.00 sec)
```

```
mysql> delete from stud where sno='2009010103';
Query OK, 1 row affected (0.64 sec)

mysql> select * from stud;
+------------+-------+------+------+--------+
| sno        | sname | ssex | sage | sdep   |
+------------+-------+------+------+--------+
| 2009010102 | 李四  | M    |   21 | 计科系 |
+------------+-------+------+------+--------+
1 row in set (0.00 sec)

mysql> select * from scsc;
+------------+------+-------+
| sno        | cno  | grade |
+------------+------+-------+
| 2009010102 | 1002 |    85 |
+------------+------+-------+
1 row in set (0.00 sec)
```

4.2.2 触发器实现CHECK功能

利用触发器实现学生的年龄只能是16~40岁，性别只能是M或F。

```
create trigger student_check before insert on student for each row
begin
  DECLARE ssex CHAR(5);
  declare sage smallint;
  set sage=new.sage;
  set ssex=new.ssex;
  if (sage>15 and sage<41) and (ssex='m' or ssex='f') then
    set @choice=1;
  else
    set @choice=0;
    insert into xxxx values (1);
  end if;
end;
```

```
mysql> select * from student;
+------------+-------+------+------+---------+
| sno        | sname | ssex | sage | sdep    |
+------------+-------+------+------+---------+
| 2009080101 | mark  | m    |   20 |         |
| 2009080102 | mary  | f    |   20 | compute |
| 2009080103 | john  | m    |   17 | math    |
+------------+-------+------+------+---------+
3 rows in set (0.00 sec)
```

```
mysql> insert into student values('2009080104','nike','m',15,'compute');
ERROR 1146 (42S02): Table 'st.xxx' doesn't exist
mysql> insert into student values('2009080104','nike','k',20,'compute');
ERROR 1146 (42S02): Table 'st.xxx' doesn't exist
```

说明：由于MYSQL数据库中没有实现UNDO的功能，同时CHECK用户自定义约束也没有实现。所以通过触发器，在进行相关操作时导致违规而不能完成来实现。

4.2.3 查看触发器

- 查看系统中所有的触发器

  SHOW GRIGGERS;

- 查看MYSQL系统数据库information_schema中的TRIGGERS表

  select * from triggers where trigger_name='stud_update' \G

4.2.3 删除触发器

- 删除语法

DROP TRIGGER [*schema_name.*]*trigger_name*

说明：从MySQL 5.0.10之前的MySQL版本升级到5.0.10或更高版本时（包括所有的MySQL 5.1版本），必须在升级之前舍弃所有的触发程序，并在随后重新创建它们，否则，在升级之后DROP TRIGGER不工作。

本章内容小结

- 4.1 存储过程和函数
  - 4.1.1 创建与修改存储过程或函数
  - 4.1.2 删除存储过程或函数
  - 4.1.3 查看存储过程或者函数
  - 4.1.4 使用变量
  - 4.1.5 定义条件和处理
  - 4.1.6 光标的使用
  - 4.1.7 流程控制语句
- 4.2 触发器
  - 4.2.1 创建触发器
  - 4.2.2 触发器实现级联、CHECK
  - 4.2.3 删除与查看触发器