MySQL 5.0 触发器

MySQL 5.0 新特性系列 C 第 2 部分

MySQL 技术白皮书

Peter Gulutzan

March, 2005

翻译：陈朋奕

毕业于西安电子科技大学

现泛华讯电脑技术

Table of Contents

Introduction

本书是为需要了解5.0版本新特性的MySQL老用户而写的。简单的来说介绍了“存储过程、触发器、视图、信息架构视图”，这是介绍MySQL 5.0新特性丛书的第一集。希望这本书能像内行专家那样与您进行对话，用简单的问题、例子让你学到需要的知识。为了达到这样的目的，我会从每一个细节开始慢慢的为大家建立概念，最后会给大家展示较大的实用用例，在学习之前也许大家会认为这个用例很难，但是只要跟着课程去学，相信很快就能掌握。

Conventions and Styles约定和编程风格

每次我想要演示实际代码时，我会对mysql客户端的屏幕就出现的代码进行调整，将字体改成Courier，使他们看起来与普通文本不一样（让大家区别程序代码和正文）。在这里举个例子：

```
mysql> DROP FUNCTION f;
Query OK, 0 rows affected (0.00 sec)
```

如果实例比较大，则需要在某些行和段落间加注释，同时我会用将“<--”符号放在页面的右边以表示强调。例如：

```
mysql> CREATE PROCEDURE p ()
    -> BEGIN
    ->    /* This procedure does nothing */                <--
    -> END;//
Query OK, 0 rows affected (0.00 sec)
```

有时候我会将例子中的"mysql>"和"->"这些系统显示去掉，你可以直接将代码复制到mysql客户端程序中（如果你现在所读的不是电子版的，可以在mysql.com网站下载相关脚本）

所以的例子都已经在Suse 9.2 Linux、Mysql 5.0.3公共版上测试通过。在您阅读本书的时候，Mysql已经有更高的版本，同时能支持更多OS了，包括Windows，Sparc，HP-UX。因此这里的例子将能正常的运行在您的电脑上。但如果运行仍然出现故障，可以咨询你认识的资深Mysql用户，这样就能得到比较好的支持和帮助。

Why Triggers 为什么要用触发器

我们在MySQL 5.0中包含对触发器的支持是由于以下原因:

- MySQL早期版本的用户长期有需要触发器的要求。
- 我们曾经许诺支持所有ANSI 标准的特性。
- 您可以使用它来检查或预防坏的数据进入数据库。
- 您可以改变或者取消INSERT, UPDATE以及DELETE语句。
- 您可以在一个会话中监视数据改变的动作。

在这里我假定大家都读过“MySQL 新特性”丛书的第一集——“MySQL 存储过程”，那么大家都应该知道MySQL至此存储过程和函数，那是很重要的知识，因为在触发器中你可以使用在函数中使用的语句。特别举个例子：

复合语句 (BEGIN / END) 是合法的.
流控制（Flow-of-control）语句 (IF, CASE, WHILE, LOOP, WHILE, REPEAT, LEAVE, ITERATE) 也是合法的.
变量声明 (DECLARE) 以及指派(SET) 是合法的.
允许条件声明.
异常处理声明也是允许的.

但是在这里要记住函数有受限条件: 不能在函数中访问表. 因此在函数中使用以下语句是非法的。

ALTER 'CACHE INDEX' CALL COMMIT CREATE DELETE
DROP 'FLUSH PRIVILEGES' GRANT INSERT KILL
LOCK OPTIMIZE REPAIR REPLACE REVOKE
ROLLBACK SAVEPOINT 'SELECT FROM table'
'SET system variable' 'SET TRANSACTION'
SHOW 'START TRANSACTION' TRUNCATE UPDATE

在触发器中也有完全一样的限制.

触发器相对而言比较新，因此会有（bugs）缺陷. 所以我在这里给大家警告，就像我在存储过程书中所说那样. 不要在含有重要数据的数据库中使用这个触发器，如果需要的话在一些以测试为目的的数据库上使用，同时在你对表创建触发器时确认这些数据库是默认的。

Syntax 语法

1. Syntax: Name 语法：命名规则

```
CREATE TRIGGER <触发器名称>                              <--
  { BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }
  ON <表名称>
  FOR EACH ROW
  <触发器SQL语句>
```

触发器必须有名字，最多64个字符，可能后面会附有分隔符. 它和MySQL中其他对象的命名方式基本相象.

这里我有个习惯：就是用表的名字＋'_'＋触发器类型的缩写. 因此如果是表 t26，触发器是在事件 UPDATE （参考下面的点（2）和（3））之前（BEFORE）的，那么它的名字就是t26_bu 。

2. Syntax: Time 语法：触发时间

```
CREATE TRIGGER <触发器名称>
  { BEFORE | AFTER }                                    <--
  { INSERT | UPDATE | DELETE }
  ON <表名称>
  FOR EACH ROW
  <触发的SQL语句>
```

触发器有执行的时间设置：可以设置为事件发生前或后。

3. Syntax: Event语法：事件

```
CREATE TRIGGER <触发器名称>
  { BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }                          <--
  ON <表名称>
  FOR EACH ROW
  <触发的SQL语句>
```

同样也能设定触发的事件：它们可以在执行insert、update或delete的过程中触发。

4. Syntax: Table 语法：表

```
CREATE TRIGGER <触发器名称>
  { BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }
  ON <表名称>                                            <--
  FOR EACH ROW
  <触发的SQL语句>
```

触发器是属于某一个表的: 当在这个表上执行插入、更新或删除操作的时候就导致触发器的激活. 我们不能给同一张表的同一个事件安排两个触发器。

5. Syntax: Granularity 语法：（步长）触发间隔

```
CREATE TRIGGER <触发器名称>
  { BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }
  ON <表名称>
  FOR EACH ROW                                           <--
  <触发的SQL语句>
```

触发器的执行间隔：FOR EACH ROW 子句通知触发器每隔一行执行一次动作，而不是对整个表执行一次。

6. Syntax: Statement 语法：语句

```
CREATE TRIGGER <触发器名称>
  { BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }
  ON <表名称>
  FOR EACH ROW
  <触发的SQL语句>                                    <--
```

触发器包含所要触发的SQL语句：这里的语句可以是任何合法的语句，包括复合语句，但是这里的语句受的限制和函数的一样。

Privileges权限

你必须拥有相当大的权限才能创建触发器（CREATE TRIGGER）。如果你已经是Root用户，那么就足够了。这跟SQL的标准有所不同，我也希望能尽快改成标准的。

因此在下一个版本的MySQL中，你完全有可能看到有一种叫做CREATE TRIGGER的新权限。然后通过这样的方法赋予：

```
GRANT CREATE TRIGGER ON <表名称> TO <用户或用户列表>;
```

也可以通过这样收回权限：

```
REVOKE CREATE TRIGGER ON <表名称> FROM <用户或用户列表>;
```

Referring to OLD and NEW columns 关于旧的和新创建的列的标识

在触发器的SQL语句中，你可以关联表中的任意列。但你不能仅仅使用列的名称去标识，那会使系统混淆，因为那里可能会有列的新名（这可能正是你要修改的，你的动作可能正是要修改列名），还有列的旧名存在。因此你必须用这样的语法来标识：
"NEW . column_name" 或者 "OLD . column_name". 这样在技术上处理（NEW | OLD . column_name）新和旧的列名属于创建了过渡变量（"transition variables"）。

对于INSERT语句, 只有NEW是合法的；对于DELETE语句，只有OLD才合法；而UPDATE语句可以在和NEW以及OLD同时使用。下面是一个UPDATE中同时使用NEW和OLD的例子。

```
CREATE TRIGGER t21_au
BEFORE UPDATE ON t22
FOR EACH ROW
BEGIN
  SET @old = OLD . s1;
  SET @new = NEW.s1;
END;//
```

现在如果t21表中的s1列的值是55，那么执行了"UPDATE t21 SET s1 = s1 + 1"之后@old的值会变成55，而@new的值将会变成56。

Example of CREATE and INSERT　　CREATE和INSERT的例子

CREATE table with trigger 创建有触发器的表

这里所有的例程中我都假定大家的分隔符已经设置成//（DELIMITER //）。

```
CREATE TABLE t22 (s1 INTEGER)//

CREATE TRIGGER t22_bi
BEFORE INSERT ON t22
FOR EACH ROW
BEGIN
  SET @x = 'Trigger was activated!';
  SET NEW.s1 = 55;
END;//
```

在最开始我创建了一个名字为t22的表，然后在表t22上创建了一个触发器t22_bi，当我们要向表中的行插入时，触发器就会被激活，执行将s1列的值改为55的动作。

INSERT on table w ith a trigger 使用触发器执行插入动作

```
mysql> INSERT INTO t22 VALUES (1)//
```

让我们看如果向表t2中插入一行数据触发器对应的表会怎么样？

这里的插入的动作是很常见的，我们不需要触发器的权限来执行它。甚至不需要知道是否有触发器关联。

```
mysql> SELECT @x, t22.* FROM t22//
+-----------------------+------+
| @x                    | s1   |
+-----------------------+------+
| Trigger was activated! |   55 |
+-----------------------+------+
1 row in set (0.00 sec)
```

大家可以看到INSERT动作之后的结果，和我们预期的一样，x标记被改动了，同时这里插入的数据不是我们开始输入的插入数据，而是触发器自己的数据。

Example of a "check" constraint “check”完整性约束例子

What's a "check" constraint　什么是“check”约束

在标准的SQL语言中，我们可以在（CREATE TABLE）创建表的过程中使用"CHECK (condition)"，例如：

```
CREATE TABLE t25
  (s1 INT, s2 CHAR(5), PRIMARY KEY (s1),
  CHECK (LEFT(s2,1)='A'))
  ENGINE=INNODB;
```

这里CHECK的意思是“当s2列的最左边的字符不是’A’时，insert和update语句都会非法”，MySQL的视图不支持CHECK，我个人是很希望它能支持的。但如果你很需要在表中使用这样的功能，我建议大家使用触发器来实现。

```
CREATE TABLE t25
(s1 INT, s2 CHAR(5),
PRIMARY KEY (s1))
ENGINE=INNODB//

CREATE TRIGGER t25_bi
BEFORE INSERT ON t25
FOR EACH ROW
IF LEFT(NEW.s2,1)<>'A' THEN SET NEW.s1=0; END IF;//

CREATE TRIGGER t25_bu
BEFORE UPDATE ON t25
FOR EACH ROW
IF LEFT(NEW.s2,1)<>'A' THEN SET NEW.s1=0; END IF;//
```

我只需要使用BEFORE INSERT和BEFORE UPDATE语句就行了，删除了触发器不会对表有影响，同时AFTER的触发器也不能修改NEW的过程变量（transition variables）。为了激活触发器，我执行了向表中的行插入s1＝0的数据，之后只要执行符合LEFT(s2,1)　<> 'A'条件的动作都会失败：

```
INSERT INTO t25 VALUES (0,'a') /* priming the pump */    //
INSERT INTO t25 VALUES (5,'b') /* gets error '23000' */ //
```

Don't Believe The Old MySQL Manual　该抛弃旧的MySQL的手册了

我在这里警告大家不要相信过去的MySQL手册中所说的了。我们已经去掉了关于触发器的错误的语句，但是仍旧有很多旧版本的手册在网上，举个例子，这是一个德国的Url上的：
http://dev.mysql.com/doc/mysql/de/ANSI_diff_Triggers.html.

这个手册上说触发器就是存储过程，忘掉吧，你也已经看见了，触发器就是触发器，而存储过程还是存储过程。

手册上还说触发器可以从其他表上来删除，或者是当你删除一个事务的时候激发，无论他说的是什么意思，忘掉吧，MySQL不会去实现这些的。

最后关于说使用触发器会对查询速度产生影响的说法也是错的，触发器不会对查询产生任何影响。

Bugs

（不好的东西就不翻译了）On December 14 2004, I did an "Advanced Search" in http://bugs.mysql.com for 'trigger' or 'triggers', I found that there were 17 active bugs as of that date. Of course they might disappear before you read this, but just in case they haven't, I'll mention the important ones. If they're still there, you'll have to work around them when you're trying triggers.

Bug#5859　DROP TABLE does not drop triggers.（ 删除表的时候没有自动删除触发器）
When you drop a table, dropping the table's triggers should be automatic.

Bug#5892 Triggers have the wrong namespace. （触发器的命名空间有错，你必须在前面加上表的名字才能删除触发器，下面是例子）

You have to say "DROP TRIGGER <table name> . <trigger name>".

The correct way is "DROP TRIGGER <trigger name>".

Bug#5894 Triggers with altered tables cause corrupt databases. （触发器对表的改变可能会造成数据库数据被破坏）

Do not alter a table that has a trigger on it, until you know this is fixed.

Conclusion 最后

到了书的最后，我认为不需要给大家复习或者是重温一下了，因为相信大家可以很轻松的记住上面所说的。

如果你喜欢这本书，那么你可以去找“MySQL 5.0 系列”其他书，较早的一本是关于“存储过程”的，下一本将会是关于“视图”的。

感谢您的关注，如果您对书有什么建议，可以在MySQL论坛上发言，下面是地址：http://forums.mysql.com

About MySQL（这里不需要翻译了）

MySQL AB develops and supports a family of high performance, affordable database servers and tools. The company's flagship product is MySQL, the world's most popular open source database, with more than six million active installations. Many of the world's largest organizations, including Yahoo!, Sabre Holdings, The Associated Press, Suzuki and NASA are realizing significant cost savings by using MySQL to power high-volume Web sites, business-critical enterprise applications and packaged software.

With headquarters in Sweden and the United States 亻C and operations around the world 亻C MySQL AB supports both open source values and corporate customers' needs in a profitable, sustainable business. For more information about MySQL, please visit www.mysql.com.