



下载APP



17 | 触发器：如何让数据修改自动触发关联操作，确保数据一致性？

2021-04-17 朱晓峰

MySQL 必知必会

[进入课程 >](#)**讲述：朱晓峰**

时长 12:18 大小 11.28M



你好，我是朱晓峰。今天，我来和你聊一聊触发器。

在实际开发中，我们经常会遇到这样的情况：有 2 个或者多个相互关联的表，如商品信息和库存信息分别存放在 2 个不同的数据表中，我们在添加一条新商品记录的时候，为了保证数据的完整性，必须同时在库存表中添加一条库存记录。

这样一来，我们就必须把这两个关联的操作步骤写到程序里面，而且要用事务包裹起来，确保这两个操作成为一个原子操作，要么全部执行，要么全部不执行。要是遇到特殊情况，可能还需要对数据进行手动维护，这样就很容易忘记其中的一步，导致数据缺失。



这个时候，其实咱们可以使用触发器。你可以创建一个触发器，让商品信息数据的插入操作自动触发库存数据的插入操作。这样一来，就不用担心因为忘记添加库存数据而导致的

数据缺失了。


听上去好像很不错，那触发器到底怎么使用呢？接下来，我就重点给你聊聊。我会先给你讲解创建、查看和删除触发器的具体操作，然后借助一个案例带你实战一下。

如何操作触发器？

首先，咱们来学习下触发器的基本操作。

创建触发器

创建触发器的语法结构是：

 复制代码

```
1 CREATE TRIGGER 触发器名称 {BEFORE|AFTER} {INSERT|UPDATE|DELETE}
2 ON 表名 FOR EACH ROW 表达式;
```

在创建时，你一定要注意触发器的三个要素。

表名：表示触发器监控的对象。

INSERT|UPDATE|DELETE：表示触发的事件。INSERT 表示插入记录时触发；UPDATE 表示更新记录时触发；DELETE 表示删除记录时触发。


BEFORE|AFTER：表示触发的时间。BEFORE 表示在事件之前触发；AFTER 表示在事件之后触发。

只有把这三个要素定义好，才能正确使用触发器。

创建好触发器之后，咱们还要知道触发器是不是创建成功了。怎么查看呢？我来介绍下。

查看触发器


查看触发器的语句是：

 复制代码

```
1 SHOW TRIGGERS\G;
```

删除触发器

删除触发器很简单，你只要知道语法结构就可以了：

 复制代码

```
1 DROP TRIGGER 触发器名称;
```

知道了触发器的操作方法，接下来咱们就借助超市项目的实际案例，在真实的场景中实战一下，毕竟，实战是掌握操作的最好方法。

案例讲解

超市项目实际实施过程中，客户经常要查询储值余额变动的明细，但是，查询会员消费流水时，存在数据汇总不及时、查询速度比较慢的问题。这时，我们就想到用触发器，及时把会员储值金额的变化信息记录到一个专门的表中。

我先用咱们熟悉的 SQL 语句来实现记录储值金额变动的操作，后面再带你使用触发器来操作。通过两种操作的对比，你就能更好地理解，在什么情况下，触发器能够比普通的 SQL 语句更加简洁高效，从而帮助你用好触发器这个工具，提高开发的能力。

下面我就借助具体数据，来详细说明一下。这里我们需要用到会员信息表（demo.membermaster）和会员储值历史表（demo.deposithist）。

会员信息表：

memberid (会员编号)	cardno (会员卡号)	membername (会员名称)	address (地址)	phone (电话)	memberdeposit (储值金额)
1	10000001	张三	北京	13812345678	100
2	10000002	李四	天津	18512345678	200

会员储值历史表：

id (编号)	memberid (会员编号)	transdate (变动时间)	oldvalue (变动前金额)	newvalue (变动后金额)	changedvalue (变动金额)

假如在 2020 年 12 月 20 日这一天，会员编号是 2 的会员李四，到超市的某家连锁店购买了一条烟，消费了 150 元。现在，我们用之前学过的 SQL 语句，把这个会员储值余额的变动情况记录到会员储值历史表中。

第一步，查询出编号是 2 的会员卡的储值金额是多少。我们可以用下面的代码来实现：

[复制代码](#)

```
1 mysql> SELECT memberdeposit
2 -> FROM demo.membermaster
3 -> WHERE memberid = 2;
4 +-----+
5 | memberdeposit |
6 +-----+
7 | 200.00 |
8 +-----+
9 1 row in set (0.00 sec)
```

第二步，我们把会员编号是 2 的会员的储值金额减去 150。

[复制代码](#)

```
1 mysql> UPDATE demo.membermaster
2 -> SET memberdeposit = memberdeposit - 150
3 -> WHERE memberid = 2;
4 Query OK, 1 row affected (0.01 sec)
5 Rows matched: 1 Changed: 1 Warnings: 0
```

第三步，读出会员编号是 2 的会员当前的储值金额。

[复制代码](#)

```

1 mysql> SELECT memberdeposit
2 -> FROM demo.membermaster
3 -> WHERE memberid = 2;
4 +-----+
5 | memberdeposit |
6 +-----+
7 | 50.00 |
8 +-----+
9 1 row in set (0.00 sec)

```

第四步，把会员编号和前面查询中获得的储值起始金额、储值余额和储值金额变化值，写入会员储值历史表。

[复制代码](#)

```

1 mysql> INSERT INTO demo.deposithist
2 -> (
3 -> memberid,
4 -> transdate,
5 -> oldvalue,
6 -> newvalue,
7 -> changedvalue
8 -> )
9 -> SELECT 2,NOW(),200,50,-150;
10 Query OK, 1 row affected (0.02 sec)
11 Records: 1 Duplicates: 0 Warnings: 0

```

这样，我们就完成了记录会员储值金额变动的操作。现在，我们来查询一下记录的结果：

[复制代码](#)

```

1 mysql> SELECT *
2 -> FROM demo.deposithist;
3 +-----+-----+-----+-----+-----+-----+
4 | id | memberid | transdate | oldvalue | newvalue | changedvalue |
5 +-----+-----+-----+-----+-----+-----+
6 | 1 | 2 | 2020-12-20 10:37:51 | 200.00 | 50.00 | -150.00 |
7 +-----+-----+-----+-----+-----+-----+
8 1 row in set (0.00 sec)

```


结果显示，会员编号是 2 的会员卡储值金额在 2020-12-20 10:37:51 时有变动，变动前是 200 元，变动后是 50 元，减少了 150 元。

你看这个记录会员储值金额变动的操作非常复杂，我们用了 4 步才完成。而且为了确保数据的一致性，我们还要用事务把这几个关联的操作包裹起来，这样一来，消耗的资源就比较多。

那如果用触发器来实现，效果会怎样呢？我们来实操一下。

首先，我们创建一个数据表 demo.membermaster 的触发器。每当更新表中的数据时，先触发触发器，如果发现会员储值金额有变化，就把会员编号信息、更新的时间、更新前的储值金额、更新后的储值金额和变动金额，写入会员储值历史表。然后，再执行会员信息表的更新操作。

创建触发器的代码如下所示：

 复制代码

```
1 DELIMITER //
```

```
2 CREATE TRIGGER demo.upd_membermaster BEFORE UPDATE -- 在更新前触发
```

```
3 ON demo.membermaster
```

```
4 FOR EACH ROW -- 表示每更新一条记录，触发一次
```

```
5 BEGIN -- 开始程序体
```

```
6 IF (new.memberdeposit <> old.memberdeposit) -- 如果储值金额有变化
```

```
7 THEN
```

```
8 INSERT INTO demo.deposithist
```

```
9 (
```

```
10 memberid,
```

```
11 transdate,
```

```
12 oldvalue,
```

```
13 newvalue,
```

```
14 changedvalue
```

```
15 )
```

```
16 SELECT
```

```
17 NEW.memberid,
```

```
18 NOW(),
```

```
19 OLD.memberdeposit, -- 更新前的储值金额
```

```
20 NEW.memberdeposit, -- 更新后的储值金额
```

```
21 NEW.memberdeposit-OLD.memberdeposit; -- 储值金额变化值
```

```
22 END IF;
```

```
23 END
```

```
24 //
```

```
25 DELIMITER ;
```

创建完成之后，我们查看一下触发器，看看是不是真的创建成功了：

```
1 mysql> SHOW TRIGGERS \G;
2 ***** 1. row *****
3 Trigger: upd_membermaster -- 触发器名称
4 Event: UPDATE -- 触发的事件
5 Table: membermaster -- 表名称
6 Statement: BEGIN -- 被触发时要执行的程序体
7 IF (new.memberdeposit <> old.memberdeposit) -- 储值金额变动时插入历史储值表
8 THEN
9 INSERT INTO demo.deposithist
10 (
11 memberid,
12 transdate,
13 oldvalue,
14 newvalue,
15 changedvalue
16 )
17 SELECT
18 NEW.memberid,
19 NOW(),
20 OLD.memberdeposit,
21 NEW.memberdeposit,
22 NEW.memberdeposit-OLD.memberdeposit;
23 END IF;
24 END
25 Timing: BEFORE
26 Created: 2021-04-03 15:02:48.18
27 sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
28 Definer: root@localhost
29 character_set_client: utf8mb4
30 collation_connection: utf8mb4_0900_ai_ci
31 Database Collation: utf8mb4_0900_ai_ci
32 1 row in set (0.00 sec)
```

从代码中，我们可以知道触发器的具体信息：

Trigger 表示触发器名称，这里是 upd_membermaster；

Event 表示触发事件，这里是 UPDATE，表示更新触发；

Table 表示定义触发器的数据表，这里是 membermaster；

Statement 表示触发时要执行的程序体。

看到这些信息，我们就可以确认，触发器创建成功了。

创建成功以后，我们尝试更新一下会员编号是 1 的会员的储值金额，这里假设把会员 1 的储值金额增加 10 元。简单说明一下，会员也可以把钱存到会员卡里，需要的时候进行消费，对应的就是储值金额的增加。

我们用下面的代码来完成这个操作：

[复制代码](#)

```
1 mysql> UPDATE demo.membermaster
2 -> SET memberdeposit = memberdeposit + 10
3 -> WHERE memberid = 1;
4 Query OK, 1 row affected (0.03 sec)
5 Rows matched: 1 Changed: 1 Warnings: 0
```

现在，我们来查看一下会员信息表和会员储值历史表：

[复制代码](#)


```
1 mysql> SELECT *
2 -> FROM demo.membermaster;
3 +-----+-----+-----+-----+-----+-----+
4 | memberid | cardno | membername | address | phone | memberdeposit |
5 +-----+-----+-----+-----+-----+-----+
6 | 1 | 10000001 | 张三 | 北京 | 13812345678 | 110.00 |
7 | 2 | 10000002 | 李四 | 天津 | 18512345678 | 50.00 |
8 +-----+-----+-----+-----+-----+-----+
9 2 rows in set (0.00 sec)
10
11 mysql> SELECT *
12 -> FROM demo.deposithist;
13 +----+-----+-----+-----+-----+-----+
14 | id | memberid | transdate | oldvalue | newvalue | changedvalue |
15 +----+-----+-----+-----+-----+-----+
16 | 1 | 2 | 2020-12-20 10:37:51 | 200.00 | 50.00 | -150.00 |
17 | 2 | 1 | 2020-12-20 11:32:09 | 100.00 | 110.00 | 10.00 |
18 +----+-----+-----+-----+-----+-----+
19 2 rows in set (0.01 sec)
```

结果显示，触发器正确地记录了修改会员储值金额的操作。

如果你一直跟着我进行操作，到这里，你可能会有疑问，在会员历史储值表中记录修改会员储值金额的操作，和实际修改会员信息表是 2 个操作，有没有可能一个成功，一个失败？比如说，记录修改会员储值金额的操作失败了，但是会员的储值金额被修改了。

其实，在 MySQL 中，如果触发器中的操作失败了，那么触发这个触发器的数据操作也会失败，不会出现一个成功、一个失败的情况。

我还是借助一个例子来解释下。假设我们创建了一个触发器，这个触发器的程序体中的 SQL 语句有误，比如多了一个字段。在这种情况下，触发器执行会失败，因此，数据更新的操作也不会执行。我们用下面的代码来验证一下：

 复制代码

```
1 DELIMITER //
```

```
2 CREATE TRIGGER demo.upd_membermaster BEFORE UPDATE
```

```
3 ON demo.membermaster
```

```
4 FOR EACH ROW
```

```
5 BEGIN
```

```
6 IF (new.memberdeposit <> old.memberdeposit)
```

```
7 THEN
```

```
8 INSERT INTO demo.deposithist
```

```
9 (
```

```
10 aa, -- 不存在的字段
```

```
11 memberid,
```

```
12 transdate,
```

```
13 oldvalue,
```

```
14 newvalue,
```

```
15 changedvalue
```

```
16 )
```

```
17 SELECT
```

```
18 1, -- 给不存在的字段赋值
```

```
19 NEW.memberid,
```

```
20 NOW(),
```

```
21 OLD.memberdeposit,
```

```
22 NEW.memberdeposit,
```

```
23 NEW.memberdeposit-OLD.memberdeposit;
```


```
24 END IF;
```

```
25 END
```

```
26 //
```

```
27 DELIMITER ;
```

现在，假设我们要把会员编号是 2 的会员卡的储值金额更新为 20：

 复制代码

```
1 mysql> update demo.membermaster set memberdeposit=20 where memberid = 2;
```

```
2 ERROR 1054 (42S22): Unknown column 'aa' in 'field list'
```

系统提示：因为字段“aa”不存在，导致触发器执行失败。现在我们来看看会员储值金额有没有被修改：

[复制代码](#)

```
1 mysql> select * from demo.membermaster;
2 +-----+-----+-----+-----+-----+-----+
3 | memberid | cardno | membername | address | phone | memberdeposit |
4 +-----+-----+-----+-----+-----+-----+
5 | 1 | 10000001 | 张三 | 北京 | 13812345678 | 110.00 |
6 | 2 | 10000002 | 李四 | 天津 | 18512345678 | 50.00 |
7 +-----+-----+-----+-----+-----+-----+
8 2 rows in set (0.00 sec)
```

结果显示，会员储值金额不变。这个时候，为了让应用程序知道触发器是否执行成功，我们可以通过 ROW_COUNT() 函数来发现错误：

[复制代码](#)

```
1 mysql> select row_count();
2 +-----+
3 | row_count() |
4 +-----+
5 | -1 |
6 +-----+
7 1 row in set (0.00 sec)
```

结果是 -1，说明我们可以通过 ROW_COUNT() 函数捕获到错误。

我们回顾一下：对于记录会员储值金额变化的操作，可以通过应用层发出 SQL 语句指令，或者用一个存储过程来实现。无论哪种方式，都要通过好几步相互关联的操作，而且要做成一个事务，处理过程复杂，消耗的资源也较多。如果用触发器，效率就会提高很多，消耗的资源也少。同时，还可以起到事务的类似功能，保证关联操作全部完成或全部失败。

触发器的优缺点

通过刚刚的案例，你应该感受到触发器的高效了。现在，咱们把视角拔高一下，来看看触发器具体都有什么优缺点。毕竟，知己知彼，才能百战不殆。只有非常清楚它的优缺点，你才能充分发挥它的作用。

我先来说说触发器的优点。

首先，触发器可以确保数据的完整性。这是怎么体现的呢？我来举个小例子。

假设我们用进货单头表（demo.importhead）来保存进货单的总体信息，包括进货单编号、供货商编号、仓库编号、总计进货数量、总计进货金额和验收日期。

listnumber (进货单编号)	supplierid (供货商编号)	stockid (参库编号)	quantity (总计数量)	importvalue (总计金额)	confirmationdate (验收日期)

用进货单明细表（demo.importdetails）来保存进货商品的明细，包括进货单编号、商品编号、进货数量、进货价格和进货金额。

listnumber (进货单编号)	itemnumber (商品编号)	quantity (进货数量)	importprice (进货价格)	importvalue (进货金额)

每当我们录入、删除和修改一条进货单明细数据的时候，进货单明细表里的数据就会发生变动。这个时候，在进货单头表中的总计数量和总计金额就必须重新计算，否则，进货单头表中的总计数量和总计金额就不等于进货单明细表中数量合计和金额合计了，这就是数据不一致。

为了解决这个问题，我们就可以使用触发器，规定每当进货单明细表有数据插入、修改和删除的操作时，自动触发 2 步操作：

- 1. 重新计算进货单明细表中的数量合计和金额合计；
- 2. 用第一步中计算出来的值更新进货单头表中的合计数量与合计金额。

这样一来，进货单头表中的合计数量与合计金额的值，就始终与进货单明细表中计算出来的合计数量与合计金额的值相同，数据就是一致的，不会互相矛盾。

其次，触发器可以帮助我们记录操作日志。

利用触发器，可以具体记录什么时间发生了什么。我们前面的记录修改会员储值金额的触发器，就是一个很好的例子。这对我们还原操作执行时的具体场景，更好地定位问题原因很有帮助。

另外，触发器还可以用在操作数据前，对数据进行合法性检查。

举个小例子。超市进货的时候，需要库管录入进货价格。但是，人为操作很容易犯错误，比如说在录入数量的时候，把条形码扫进去了；录入金额的时候，看串了行，录入的价格远超售价，导致账面上的巨亏.....这些都可以通过触发器，在实际插入或者更新操作之前，对相应的数据进行检查，及时提示错误，防止错误数据进入系统。

说了这么多触发器的优点，那是不是所有事件可以驱动的操作，都应该用触发器呢？要是你这么想，就掉坑里了。


下面我来说说触发器的缺点。

触发器最大的一个问题就是可读性差。

因为触发器存储在数据库中，并且由事件驱动，这就意味着触发器有可能不受应用层的控制。这对系统维护是非常有挑战的。

这是啥意思呢？我举个例子，你一看就明白了。

还是拿我们创建触发器时讲到的修改会员储值操作的那个触发器为例。如果触发器中的操作出了问题，会导致会员储值金额更新失败。我用下面的代码演示一下：

 复制代码

```
1 mysql> update demo.membermaster set memberdeposit=20 where memberid = 2;  
2 ERROR 1054 (42S22): Unknown column 'aa' in 'field list'
```


结果显示，系统提示错误，字段“aa”不存在。

这是因为，触发器中的数据插入操作多了一个字段，系统提示错误。可是，如果你不了解这个触发器，很可能会认为是更新语句本身的问题，或者是会员信息表的结构出了问题。说不定你还会给会员信息表添加一个叫“aa”的字段，试图解决这个问题，结果只能是白费力气。

另外，相关数据的变更，特别是数据表结构的变更，都可能会导致触发器出错，进而影响数据操作的正常运行。这些都会由于触发器本身的隐蔽性，影响到应用中错误原因排查的效率。

总结

今天这节课，我给你介绍了如何操作触发器。为了方便你学习，我汇总了相关的语法结构：

 复制代码

```
1 创建触发器的语法结构是
2 CREATE TRIGGER 触发器名称 {BEFORE|AFTER} {INSERT|UPDATE|DELETE}
3 ON 表名 FOR EACH ROW 表达式;
4
5 查看触发器的语句是：
6 SHOW TRIGGERS\G;
7
8 删除触发器的语法结构是：
9 DROP TRIGGER 触发器名称;
```

除此之外，我们还学习了触发器的优缺点。它的优点是可以确保数据一致性、记录操作日志和检查数据合法性。不过，它也存在可读性差，会增加系统维护的成本的缺点。在使用触发器的时候，你一定要综合考量。

最后，我还想再给你提一个小建议：**维护一个完整的数据库设计文档**。因为运维人员可能会经常变动，如果有一个完整的数据库设计文档，就可以帮助新人快速了解触发器的设计思路，从而减少错误，降低系统维护的成本。

思考题

我在课程中提到，每当在进货单明细表中插入或修改数据的时候，都要更新进货单头表中的总计数量和总计金额，这个问题可以用触发器来解决。你能不能创建一个触发器，要求是当操作人员更新进货单明细表中相关数据的时候，自动触发对进货单头表中相关数据的更改，确保数据的一致性？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，欢迎你把它分享给你的朋友或同事，我们下节课见。

提建议

更多课程推荐

深入浅出计算机组成原理

带你掌握计算机体系全貌

徐文浩

bothub 创始人



涨价倒计时 🕒

今日订阅 **¥89**，5月12日涨价至 **¥199**

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 16 | 游标：对于数据集中的记录，该怎么逐条处理？

下一篇 18 | 权限管理：如何控制数据库访问，消除安全隐患？

精选留言 (5)

[写留言](#)**危大爷危**

2021-04-25

老师能不能提供一个数据库设计文档给我们学习一下啊..

展开 ∨

**Harry**

2021-04-18

触发器和事务的根本差别在哪呢？

展开 ∨

**Harry**

2021-04-18

视图，存储过程，存储函数，触发器
这些对象都保存在数据库中
对于应用系统来说具有很大的隐蔽性
维护和管理它们还是有很高成本的

展开 ∨

**Harry**

2021-04-18

请问老师，
一个完整的数据库设计文档应该包含哪些内容呢？

展开 ∨

**lesserror**

2021-04-18

触发器实现的功能让我想到了web框架的模型事件。实现的也是模型更新、删除等前后需要执行的操作。

所以现在在代码层面处理这类多个表数据之间的数据一致性操作变得没有那么麻烦。

...

展开

