链滴

MySQL / MariaDB 触发器的创建、使用、查看、删除教程及应用场景实战案例

作者：HiJiangChuan

原文链接：https://ld246.com/article/1636741059794

来源网站：链滴

本文首发：https://kalacloud.com/blog/how-to-manage-and-use-mysql-database-triggers

触发器（Trigger）是 MySQL 中非常实用的一个功能，它可以在操作者对表进行「增删改」 之前（之后）被触发，自动执行一段事先写好的 SQL 代码。

本教程带领大家在实践中学习，你将学到触发器在实际应用场景中的重要应用。

在这个教程中，你是「卡拉云银行」的程序员，你正在搭建一套银行客户管理系统。在这套系统中，需要设置在INSERT 表之前检测操作者是否输入错误数据、在 UPDATE 时，记录操作者的行为 log 以及在DELETE 时，判断删除的信息是否符合删除规则。 这三类操作都可以使用 MySQL 触发器来实。

如果你正在数据库的基础上搭建一套数据库管理工具或企业内部工具，推荐你试试我开发的卡拉云，情见后文。

本教程将带你一起实践的案例

- BEFORE INSERT： 在插入数据前，检测插入数据是否符合业务逻辑，如不符合返回错误信息。
- AFTER INSERT： 在表 A 创建新账户后，将创建成功信息自动写入表 B 中。
- BEFORE UPDATE：在更新数据前，检测更新数据是否符合业务逻辑，如不符合返回错误信息。
- AFTER INSERT：在更新数据后，将操作行为记录在 log 中
- BEFORE DELETE：在删除数据前，检查是否有关联数据，如有，停止删除操作。
- AFTER DELETE：删除表 A 信息后，自动删除表 B 中与表 A 相关联的信息。

先决条件

在开始之前，请确保您具备以下条件：

- 一台配置好的 Ubuntu 服务器，root 账号。
- 服务器上配置好 MySQL Server（配置 MySQL 请看 MySQL安装及连接MySQL教程）
- MySQL root 账号

创建示例数据库

我们先创建一个干净的示例数据库，方便大家可以跟随本教程一起实践。我们会在这个数据库中演示 ySQL 触发器的多种工作方式。

首先，以 root 身份登录到你的 MySQL 服务器：

```
mysql -u root -p
```

出现提示时，请输入你 MySQL root 账号的密码，然后点击 ENTER 继续。看到 mysql> 提示后，运以下命令，创建 demo_kalacloud 数据库：

```
CREATE database demo_kalacloud;
```

```
Output
Query OK, 1 row affected (0.00 sec)
```

接下来，切换到新建的 demo_kalacloud 数据库：

```
USE demo_kalacloud;
```

```
Output
Database changed
```

接着创建一个 customers 表。我们使用这个表记录银行客户的信息。这个表包括 customer_id，customer_name，和level。咱们先把客户分为两个级别：BASIC和VIP。

```
create table customers(
customer_id BIGINT PRIMARY KEY,
customer_name VARCHAR(50),
level VARCHAR(50)
) ENGINE=INNODB;
```

```
Output
Query OK, 0 rows affected (0.01 sec)
```

接着，我们向 customers 表中添加一些客户记录。

```
Insert into customers (customer_id, customer_name, level )values('1','Jack Ma','BASIC');
Insert into customers (customer_id, customer_name, level )values('2','Robin Li','BASIC');
Insert into customers (customer_id, customer_name, level )values('3','Pony Ma','VIP');
```

分别运行三个 INSERT 命令后，命令行输出成功信息。

```
Output
Query OK, 1 row affected (0.01 sec)
```

我们使用 SELECT 检查一下三条信息是否已经写入表中：

Select * from customers;

下面我们创建另一个表customer_status，用于保存 customers 表中客户的备注信息。

这个表包含 customer_id 和 status_notes 字段：

Create table customer_status(customer_id BIGINT PRIMARY KEY, status_notes VARCHAR(50)) NGINE=INNODB;

然后，我们再创建一个 sales 表，这个表与 customer_id 关联。保存与客户有关的销售数据。

Create table sales(sales_id BIGINT PRIMARY KEY, customer_id BIGINT, sales_amount DOUBLE) ENGINE=INNODB;

Output
Query OK, 0 rows affected (0.01 sec)

最后一步，我们再建一个 audit_log 表，用来记录操作员操作「卡拉云银行」客户管理系统时的操作为。方便管理员在发生问题时，有 log 可查。

Create table audit_log(log_id BIGINT PRIMARY KEY AUTO_INCREMENT, sales_id BIGINT, previous_amount DOUBLE, new_amount DOUBLE, updated_by VARCHAR(50), updated_on DATETIM) ENGINE=INNODB;

Output
Query OK, 0 rows affected (0.02 sec)

至此，你作为「卡拉云银行」的程序员，已经把客户管理系统的demo_kalacloud 数据库和四张表建完成。接下来，我们将对这个管理系统的关键节点增加对应的触发器。

扩展阅读：《如何使用 MySQL 慢查询日志进行性能优化 - Profiling、mysqldumpslow 实例详解》

1.BEFORE INSERT触发器使用方法

作为严谨的银行客户管理系统，对任何写入系统的数据都应该提前检测，以防止错误的信息被写进去。

在写入前检测数据这个功能，我们可以使用BEFORE INSERT 触发器来实现。

在操作者对 sales 表中的sales_amount 字段进行写操作时，系统将在写入（INSERT）前检查数据是符合规范。

我们先来看一下，**创建触发器的基本语法。**

```
DELIMITER //
CREATE TRIGGER [触发器的名字]
[触发器执行时机] [触发器监测的对象]
ON [表名]
FOR EACH ROW [触发器主体代码]//
DELIMITER ;
```

触发器的结构包括：

- DELIMITER //：MySQL 默认分隔符是; 但在触发器中，我们使用 // 表示触发器的开始与结束。
- [触发器的名字]：这里填写触发器的名字
- [触发器执行时机]：这里设置触发器是在关键动作执行之前触发，还是执行之后触发。
- [触发器监测的对象]：触发器可以监测 INSERT、UPDATE、DELETE 的操作，当监测的命令对触发关联的表进行操作时，触发器就被激活了。
- [表名]：将这个触发器与数据库中的表进行关联，触发器定义在表上，也附着在表上，如果这个表删除了，那么这个触发器也随之被删除。
- FOR EACH ROW：这句表示只要满足触发器触发条件，触发器都会被执行，也就是说带上这个参后，触发器将监测每一行对关联表操作的代码，一旦符合条件，触发器就会被触发。
- [触发器主体代码]：这里是当满足触发条件后，被触发执行的代码主体。这里可以是一句 SQL 语句也可以是多行命令。如果是多行命令，那么这些命令要写在 BEGIN...END 之间。

注：在创建触发器主体时，还可以使用OLD和NEW 来获取 SQL 执行INSERT，UPDATE和DELETE操作前后的写入数据。这里没看明白没关系，我们将会在接下来的实践中，展开讲解。

讲到这里，大家看了一大堆云里雾里的概念，如果没看懂，也别担心。接下来进入实践环节，只要跟贴代码看返回结果，很快你就能够通透理解触发器了。

现在，我们来创建第一个触发器，BEFORE INSERT （在执行 insert 之前，执行触发器）。这个触发用于监测操作者在写入 sales 表中的 sales_amount 值时，这个值是否大于 10000 ，如果大于，那返回错误信息进行报错。

登录 MySQL Server 后，我们创建一个触发器：

```
DELIMITER //
CREATE TRIGGER validate_sales_amount
BEFORE INSERT
ON sales
FOR EACH ROW
IF NEW.sales_amount>10000 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "你输入的销售总额超过 10000 元。";
END IF//
DELIMITER ;
```

上面这段代码中，我们使用IF...THEN...END IF 来创建一个监测 INSERT 语句写入的值是否在限定的围内的触发器。

这个触发器的功能时监测 INSERT 在写入sales_amount 值时，这个新增的（NEW）值是否符合条件 > 10000）。

当操作员录入一个超过 10000 的数字，会返回如下错误信息：

```
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = '你输入的销售总额超过 10000 元。';
```

我们来试试看，看看触发器是否已启用。

我们向 sales_amount 中插入一条 11000 的值。

```
Insert into sales(sales_id, customer_id, sales_amount) values('1','1','11000');
```

命令行返回错误信息，这就是我们刚刚创建触发器时，填入的错误信息。与我们的设置一致。

下面我们 insert 一个值小于 10000 的数字：

```
Insert into  sales(sales_id, customer_id, sales_amount) values('1','1','7700');
```

输入值为 7700 小于设定的 10000 ，insert 命令执行成功。

```
Output
Query OK, 1 row affected (0.01 sec)
```

我们调出 sales 表，看看是否插入成功：

```
Select * from sales;
```

输出确认数据在表中：

![确认数据在表中](https://kalacloud.com/static/6a4d3093e730b33651a25d716fb6c765/29beb04-insert-into.png)

通过这张表，我们可以看到，7700 已经插入到表中。

刚刚我们演示了在执行 insert 命令前，检测某个值是否符合设定，接着我们来看在执行 insert 之后使用触发器将不同的值保存到不同的表中。

扩展阅读：《如何在两台服务器之间迁移 MySQL / MariaDB 数据库 阿里云腾讯云迁移案例》

2.AFTER INSERT触发器使用方法

接着我们讲解 AFTER INSERT ，触发器在监测到我们成功执行了 INSERT 命令后，再执行触发器中置好的代码。

例如：在银行账户系统中，当我们新建一个账户后，我们将创建成功信息写入对应的 customer_status 表中。

在这个案例中，你作为「卡拉云银行」的程序员，现在要创建一个AFTER INSERT触发器，在创建新户账户后，将成功信息写入customer_status 表中

要创建AFTER INSERT触发器，请输入以下命令：

```
DELIMITER //
CREATE TRIGGER customer_status_records
AFTER INSERT
ON customers
FOR EACH ROW
Insert into customer_status(customer_id, status_notes) VALUES(NEW.customer_id, '账户创建功')//
DELIMITER ;
```

Output
Query OK, 0 rows affected (0.00 sec)

这个触发器在操作者向 customers 表中 INSERT 新客户信息后，再向 customer_status 表对应的行写入成功信息。

现在我们 INSERT 一条信息，看看触发器是否已启用：

```
Insert into customers (customer_id, customer_name, level )values('4','Xing Wang','VIP');
```

Output
Query OK, 1 row affected (0.01 sec)

记录 INSERT 成功，接着我们来检查customer_status表中是否写入了对应的成功数据。

```
Select * from customer_status;
```

这里可以看到，我们向 customers 表插入了一个customer_id 为 4 的新用户，随后，触发器根据码自动向customer_status 表中也插入了一个 customer_id 为 4 的开户成功信息。

AFTER INSERT 特别适合这种状态变更的关联写入操作。比如开户、暂停、注销等各类状态变更。

到这里，触发器在INSERT执行前、后的应用，我们已经讲完了，接着我们来讲 UPDATE 触发器。

扩展阅读：《MySQL 配置文件 my.cnf / my.ini 逐行详解》

3.BEFORE UPDATE触发器使用方法

BEFORE UPDATE触发器与BEFORE INSERT 触发器非常类似，我们可以使用BEFORE UPDATE 触发在更新数据之前，先做一次业务逻辑检测，避免发生误操作。

刚刚我们创建示例数据库时，创建了两个级别的客户，VIP 和 BASIC 级别。卡拉云银行的客户一旦升至 VIP，就不能再降级至 BASIC 级别了。

我们使用 BEFORE UPDATE 来贯彻这一规则，这个触发器将在 UPDATE 语句执行之前，先判断是否降级行为，如果是，则输出报错信息。

我们来创建这个触发器：

```
DELIMITER //
CREATE TRIGGER validate_customer_level
BEFORE UPDATE
ON customers
FOR EACH ROW
IF OLD.level='VIP' THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'VIP 级别客户不能降级为普通级别客户';
END IF //
DELIMITER ;
```

我们可以使用 OLD 来获取执行 UPDATE 命令前，客户的 level 值。同样，我们使用该IF...THEN...END IF语句来对 level 值是否符合规则进行判断。

我们先来查看一下 customers 表中的数据。

```
select * from customers;
```

好，我们选一个已经是 VIP 级别的客户，对他进行降级操作，看看我们的触发器是否能够正确执行。

接下来，运行以下 SQL 命令，试试能不能将 customer_id 为 3 的 VIP 客户降级成 BASIC 客户：

```
Update customers set level='BASIC' where customer_id='3';
```

执行代码后，命令行返回错误信息：

```
卡拉云 kalacloud.com 低代码开发工具

mysql>Update customers set level='BASIC' where customer_id='3';

mysql>ERROR 1644 (45000): VIP 级别客户不能降级为普通级别客户。
```

这说明我们刚刚设置的触发器已经起作用了。

接着我们来试试，对一个BASIC级别的客户运行相同的命令，看看能不能把他升级到VIP级别：

```
Update customers set level='VIP' where customer_id='2';
```

执行成功：

```
Output
Rows matched: 1  Changed: 1  Warnings: 0
```

我们再来看一下 customers 表中的数据情况：

```
select * from customers;
```

```
卡拉云 kalacloud.com 低代码开发工具

mysql>Update customers set level='VIP' where customer_id='2';
Rows matched: 1  Changed: 1  Warnings: 0

mysql>select * from customers;
+-------------+---------------+-------+
| customer_id | customer_name | level |
+-------------+---------------+-------+
|           1 | Jack Ma       | BASIC |
|           2 | Robin Li      | VIP   |
|           3 | Pony Ma       | VIP   |
|           4 | Xing Wang     | VIP   |
+-------------+---------------+-------+
4 rows in set (0.00 sec)
```

可以看到刚才 customer_id 为 2 的 BASIC 客户已经升级为 VIP 客户。

BEFORE UPDATE 触发器用于在更新数据前进行确认，很好的守护了系统的业务规则。接着我们来看 AFTER UPDATE 在客户管理系统中的应用。

扩展阅读：《MySQL Workbench 操作 MySQL / MariaDB 数据库中文指南》

4.AFTER INSERT触发器使用方法

本节我们来演示 AFTER UPDATE 在实际中的应用。AFTER UPDATE **多用于 log 记录**，在管理系统操作者使用的环境中，管理员需要设置操作 log 记录，以便在出问题时，可以查看操作者对表编辑的作，可追根溯源。

我们先来创建一个对 sales 表操作的 log 记录触发器。

当操作者对 sales 表进行修改后，操作记录会被写入 audit_log 表中。

触发器将监测用户 ID 、更新前的销售总额、更新后的销售总额、操作者 ID、修改时间等信息，作为 l g 存入 audit_log 表中。

使用以下命令建立这个 log 记录触发器：

```
DELIMITER //
CREATE TRIGGER log_sales_updates
AFTER UPDATE
ON sales
FOR EACH ROW
Insert into audit_log(sales_id, previous_amount, new_amount, updated_by, updated_on) VALU
S (NEW.sales_id,OLD.sales_amount, NEW.sales_amount,(SELECT USER()), NOW() )//
DELIMITER ;
```

当操作者对 sales 表中的一条客户信息进行 UPDATE 操作时，触发器会在UPDATE操作之后，将操行为记录在 audit_log 中。包括 sales_id ，修改 sales_amount 值的前后变化。

销售总额的变化是审计的关键数据，所以要把它记录在 audit_log 中。使用OLD 来获取更新前的 sale _amount 值，使用 NEW 来获取更新后的值。

另外我们还要记录修改 sales 表的操作者信息及操作时间。

你可以使用 SELECT USER() 来检测当前操作用户的账号，用 NOW() 语句抓去当前服务器日期和时 。

为了测试这个触发器，我们先在 sales 表中创建一条信息记录：

```
Insert into sales(sales_id, customer_id, sales_amount) values('5', '2','8000');
```

```
Output
Query OK, 1 row affected (0.00 sec)
```

接下来，我们来更新这条记录：

```
Update sales set sales_amount='9000' where sales_id='5';
```

您将看到以下输出：

```
Output
Rows matched: 1  Changed: 1  Warnings: 0
```

理论上，我们更新了 sales 表后，触发器应该触发了操作，将我们刚刚的修改记录到了audit_log 表。我们用以下命令，看看audit_log 表中是否已经有记录了。

```
Select * from audit_log;
```

如下表，触发器更新了audit_log 表，表中包含了sales_amount 更新前的旧值和更新后的新值。

至此，使用 AFTER UPDATE 制作的 log 自动记录触发器就完成了。

下一节，我们来学习 DELETE 相关的触发器。

扩展阅读：《如何查看 MySQL 数据库、表、索引容量大小？找到占用空间最大的表》

5.BEFORE DELETE触发器使用方法

BEFORE DELETE触发器会在DELETE语句执行之前调用。

这些类型的触发器通常用于在不同的相关表上强制执行参照完整性。

BEFORE DELETE 的应用场景通常是确保有关联的数据不被错误的误删除掉。

例如：sales 表通过customer_id 与customers表相关联。如果操作者删除了customers 表中的一条据，那么 sales 表中某些数据就失去了关联线索。

为了避免这种情况的发生，我们需要创建一个 BEFORE DELETE触发器，防止记录被误删除。

```
DELIMITER //
CREATE TRIGGER validate_related_records
BEFORE DELETE
ON customers
FOR EACH ROW
IF OLD.customer_id in (select customer_id from sales) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = '这位客户有相关联的销售记录，不能删除。';
END IF//
DELIMITER ;
```

现在，我们试着删除有销售关联信息的客户：

```
Delete from customers where customer_id='2';
```

所以，你会看到以下输出：

这个触发器做到了先检测 sales 是否与正要被删除的 customers 表中的数据有关联，防止有关联信的数据被误删除。

不过有时候，我们需要删除主数据后，再让系统自动帮我们删除与之相关联的其他所有数据。这时，们就要用到 AFTER DELETE 这个触发器了。

扩展阅读：《在 MySQL 中 DATETIME 和 TIMESTAMP 时间类型的区别及使用场景 - 实战案例讲解》

6.AFTER DELETE触发器使用方法

接着说说 AFTER DELETE ，一旦记录被成功删除，这个触发器就会被激活。

这个触发器在实际场景用的应用也比较广泛。比如银行系统中的升级降级操作，当客户花掉自己的账积分后，激活触发器，触发器可以判断剩余积分是否满足客户当前等级，如果不满足，自动做降级操。

AFTER DELETE触发器的另一个用途是在删除主表中的数据后，与这个主表关联的数据，一起自动删。

我们来看一下这个触发器如何创建：

```
DELIMITER //
CREATE TRIGGER delete_related_info
AFTER DELETE
ON sales
FOR EACH ROW
Delete from customers where customer_id=OLD.customer_id;//
DELIMITER ;
```

接下来，我们来试试这个触发器。删除销售记录中 customer_id 为 2 的销售记录：

```
Delete from sales where customer_id='2';
```

```
Output
Query OK, 1 row affected (0.00 sec)
```

接着我们检查以下 customers 表中的关联信息是否一起自动删除：

```
Select * from customers where customer_id='2';
```

命令行会返回 Empty Set 的结果，我们刚刚删除了 sales 表中的信息后，customers 表中的关联信也被一起删除了。

以上就是 MySQL 触发器的六种使用方式和对应的场景。

扩展阅读：《最好用的 10 款 MySQL / MariaDB 管理工具横向测评 - 免费和付费到底怎么选?》

7.查看触发器

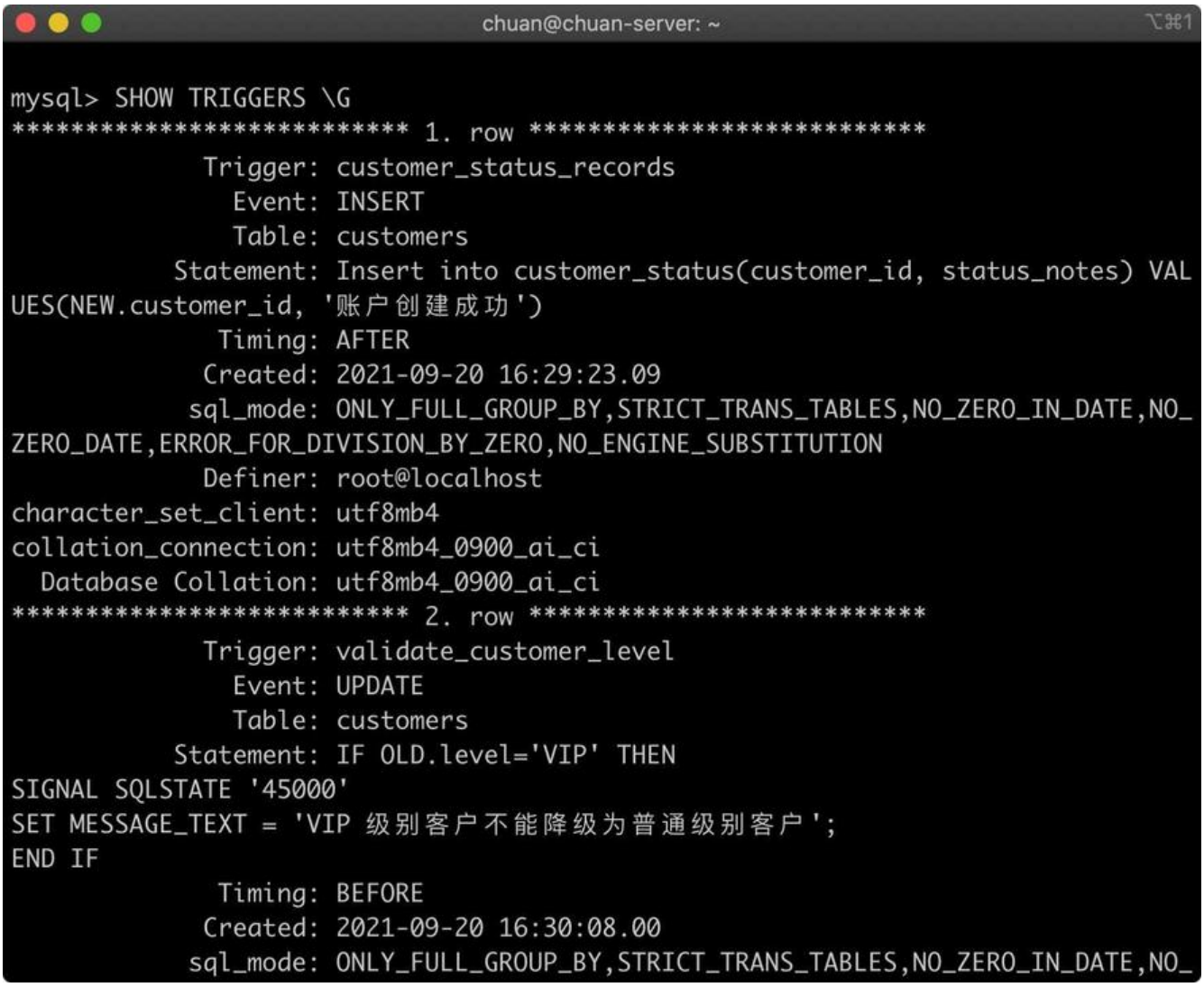
（1）直接查看触发器

当我们想查看数据库中的触发器有哪些时，可用以下命令：

SHOW TRIGGERS;

后面加上 \G 是触发器列表竖排列：

SHOW TRIGGERS \G



刚刚我们创建的触发器都罗列在这个列表当中了。

（2）在 triggers 表中查看触发器信息

在 MySQL Server 中，数据库 information_schema 的 triggers 表中存着所有触发器的信息。所有们可以通过 SELECT 来查看。

```
SELECT * FROM information_schema.triggers WHERE trigger_name= '触发器名称';
```

当然，也可以不指定触发器名称，来查看所有。

```
SELECT * FROM information_schema.triggers \G
```

扩展阅读：《如何在 MySQL / MariaDB 中查询数据库中带有某个字段/列名的所有表名》

8.删除触发器

最后，咱们来说说如何删除触发器。删除命令也很简单，Drop trigger 触发器名字 即可。

```
Drop trigger [触发器名称];
```

例如，咱们把刚刚创建的最后一个触发器删掉：

```
Drop trigger delete_related_info;

Output
Query OK, 0 rows affected (0.00 sec)
```

特别提示：我们不能对已经创建好的触发器进行修改。如果你想修改，只能先删除，再重新创建。

扩展阅读：《MySQL / MariaDB 中如何存储图片 BLOB 数据类型详解》

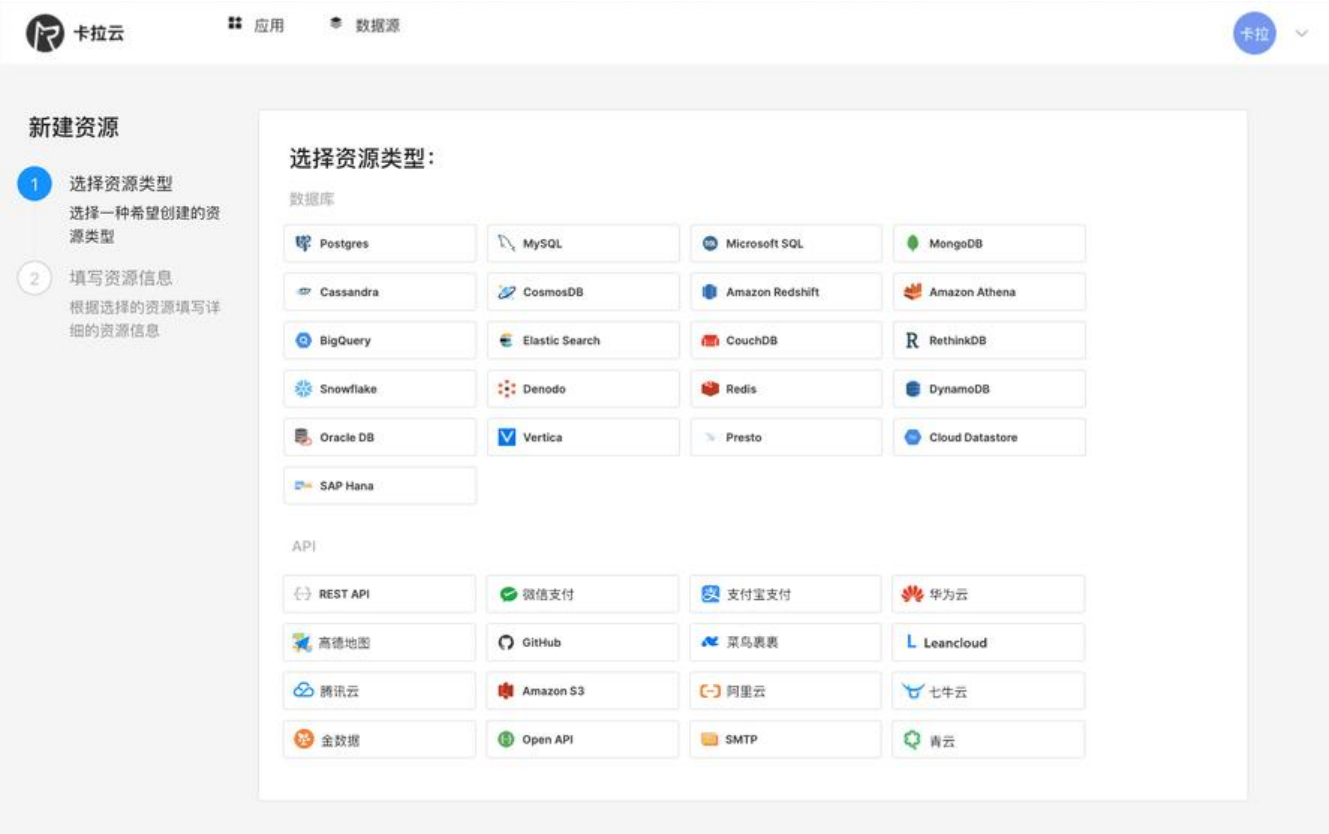
9.总结

在本教程中，我们展示了触发器的六种形式，即在INSERT、DELETE、UPDATE 执行前或后执行触发，以及对应的六个实战案例。

- BEFORE INSERT： 在插入数据前，检测插入数据是否符合业务逻辑，如不符合返回错误信息。
- AFTER INSERT： 在表 A 创建新账户后，将创建成功信息自动写入表 B 中。
- BEFORE UPDATE：在更新数据前，检测更新数据是否符合业务逻辑，如不符合返回错误信息。
- AFTER INSERT：在更新数据后，将操作行为记录在 log 中
- BEFORE DELETE：在删除数据前，检查是否有关联数据，如有，停止删除操作。
- AFTER DELETE：删除表 A 信息后，自动删除表 B 中与表 A 相关联的信息。

接着推荐一下卡拉云，只要你会写 MySQL ，就能使用卡拉云搭建自己的数据工具，比如，数据看板企业 CRM、ERP，权限管理后台，对账系统等。

卡拉云是新一代低代码开发工具，免安装部署，可一键接入包括 MySQL 在内的常见数据库及 API。根据自己的工作流，定制开发。无需繁琐的前端开发，只需要简单拖拽，即可快速搭建企业内部工具。**月的开发工作量，使用卡拉云后可缩减至数天，欢迎免费试用卡拉云。**

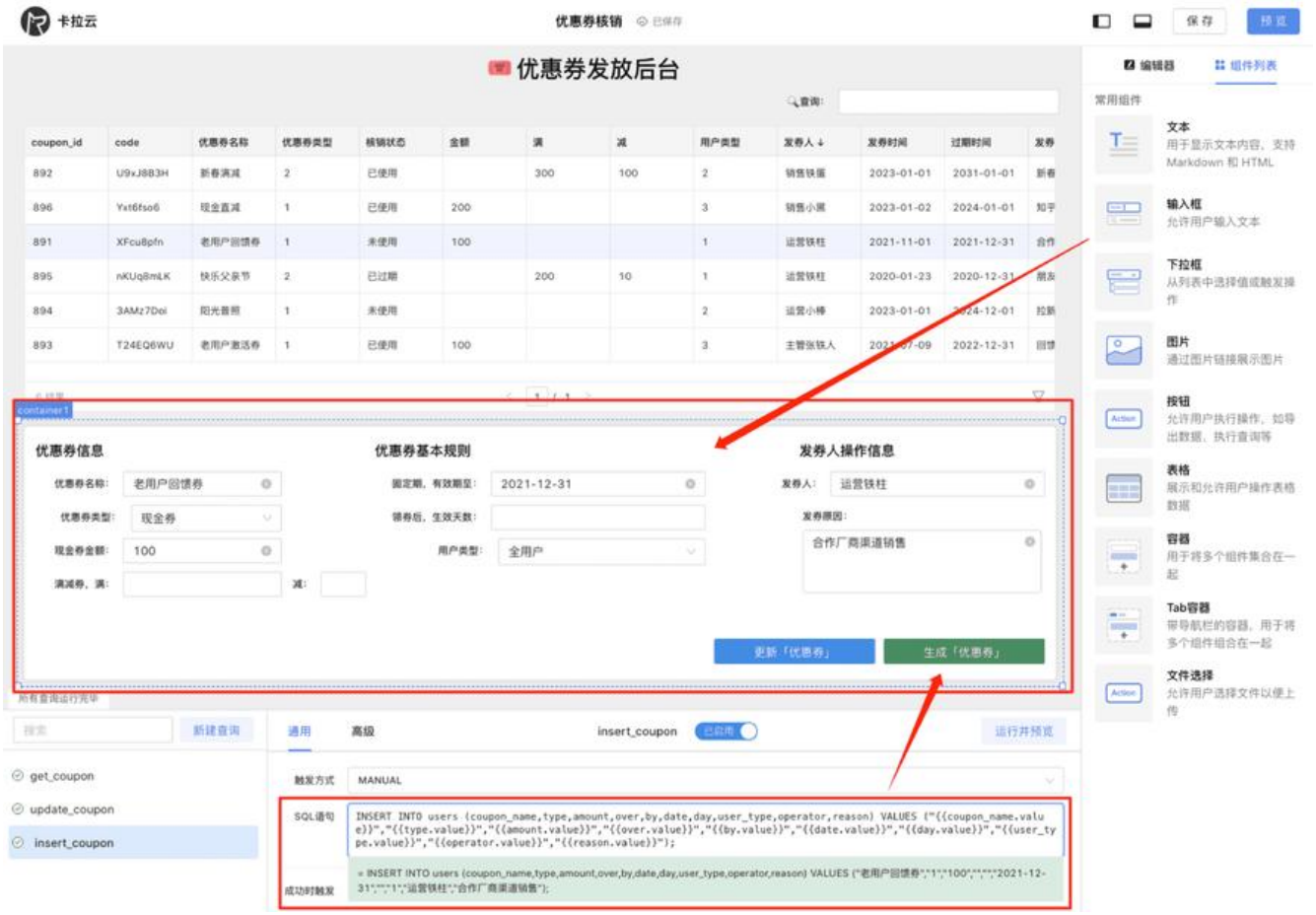


卡拉云可一键接入常见的数据库及 API

卡拉云可根据公司工作流需求，轻松搭建数据看板或其他内部工具，并且可一键分享给组内的小伙伴。

![卡拉云5分钟搭建企业内部工具(https://kalacloud.com/5400a60956e16d655e0297c5d6e5a8d2/8-kalacloud-gif.gif)

下图为使用卡拉云在 5 分钟内搭建的「优惠券发放核销」后台，仅需要简单拖拽即可快速生成前端组，只要会写 SQL，便可搭建一套趁手的数据库工具。**欢迎免费试用卡拉云。**



希望本教程对你有所帮助。更多有关 MySQL 教程，欢迎访问卡拉云查看更多。

有关 MySQL 教程，可继续拓展学习：

- 如何远程连接 MySQL 数据库，阿里云腾讯云外网连接教程
- 如何在 MySQL / MariaDB 中导入导出数据，导入导出数据库文件、Excel、CSV
- 如何在两台服务器之间迁移 MySQL 数据库 阿里云腾讯云迁移案例
- MySQL 重置自增 ID (AUTO_INCREMENT)教程 - 完美保留表数据的终极解决方案