14丨什么是事务处理，如何使用COMMIT和ROLLBACK进行操作?

我们知道在 MySQL 5.5 版本之前，默认的存储引擎是 MyISAM，在 5.5 版本之后默认存储引擎是 InnoDB。InnoDB 和 MyISAM 区别之一就是 InnoDB 支持事务，也可以说这是 InnoDB 取代 MyISAM 的重要原因。那么什么是事务呢？事务的英文是 transaction，从英文中你也能看出来它是进行一次处理的基本单元，要么完全执行，要么都不执行。

这么讲，你可能觉得有些抽象，我换一种方式讲。

不知道你是否遇到过这样的情况，你去家门口的小卖铺买东西，已经交了钱，但是老板比较忙接了个电话，忘记你是否交过钱，然后让你重新付款，这时你还要找之前的付款记录证明你已经完成了付款。

实际上如果我们线下的交易也能支持事务（满足事务的特性），就不会出现交了钱却拿不到商品的烦恼了，同样，对于小卖铺的老板来说，也不存在给出了商品但没有收到款的风险。总之，事务保证了一次处理的完整性，也保证了数据库中的数据一致性。它是一种高级的数据处理方式，如果我们在增加、删除、修改的时候某一个环节出了错，它允许我们回滚还原。正是因为这个特点，事务非常适合应用在安全性高的场景里，比如金融行业等。

我们今天就来学习下 SQL 中的事务。今天的课程你将重点掌握以下的内容：

1. 事务的特性是什么？如何理解它们?
2. 如何对事务进行控制？控制的命令都有哪些?
3. 为什么我们执行 COMMIT、ROLLBACK 这些命令的时候，有时会成功，有时会失败?

事务的特性：ACID

我刚才提到了事务的特性：要么完全执行，要么都不执行。不过要对事务进行更深一步的理解，还要从事务的 4 个特性说起，这 4 个特性用英文字母来表达就是 ACID。

1. A，也就是原子性（Atomicity）。原子的概念就是不可分割，你可以把它理解为组成物质的基本单位，也是我们进行数据处理操作的基本单位。
2. C，就是一致性（Consistency）。一致性指的就是数据库在进行事务操作后，会由原来的一致状态，变成另一种一致的状态。也就是说当事务提交后，或者当事务发生回滚

后，数据库的完整性约束不能被破坏。

3. I，就是隔离性（Isolation）。它指的是每个事务都是彼此独立的，不会受到其他事务的执行影响。也就是说一个事务在提交之前，对其他事务都是不可见的。
4. 最后一个 D，指的是持久性（Durability）。事务提交之后对数据的修改是持久性的，即使在系统出故障的情况下，比如系统崩溃或者存储介质发生故障，数据的修改依然是有效的。因为当事务完成，数据库的日志就会被更新，这时可以通过日志，让系统恢复到最后一次成功的更新状态。

ACID 可以说是事务的四大特性，在这四个特性中，原子性是基础，隔离性是手段，一致性是约束条件，而持久性是我们的目的。原子性和隔离性比较好理解，这里我讲下对一致性的理解（国内很多网站上对一致性的阐述有误，具体你可以参考 Wikipedia 对Consistency 的阐述）。

我之前讲到过数据表的 7 种常见约束（对应 04 篇）。这里指的一致性本身是由具体的业务定义的，也就是说，任何写入数据库中的数据都需要满足我们事先定义的约束规则。

比如说，在数据表中我们将姓名字段设置为唯一性约束，这时当事务进行提交或者事务发生回滚的时候，如果数据表中的姓名非唯一，就破坏了事务的一致性要求。所以说，事务操作会让数据表的状态变成另一种一致的状态，如果事务中的某个操作失败了，系统就会自动撤销当前正在执行的事务，返回到事务操作之前的状态。

事务的另一个特点就是持久性，持久性是通过事务日志来保证的。日志包括了回滚日志和重做日志。当我们通过事务对数据进行修改的时候，首先会将数据库的变化信息记录到重做日志中，然后再对数据库中对应的行进行修改。这样做的好处是，即使数据库系统崩溃，数据库重启后也能找到没有更新到数据库系统中的重做日志，重新执行，从而使事务具有持久性。

事务的控制

当我们了解了事务的特性后，再来看下如何使用事务。我们知道 Oracle 是支持事务的，而在 MySQL 中，则需要选择适合的存储引擎才可以支持事务。如果你使用的是 MySQL，可以通过 SHOW ENGINES 命令来查看当前 MySQL 支持的存储引擎都有哪些，以及这些存储引擎是否支持事务。

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.01 sec)
```

你能看出在 MySQL 中，InnoDB 是支持事务的，而 MyISAM 存储引擎不支持事务。

看到这里，我们已经对事务有了一定的了解，现在我们再来看下事务的常用控制语句都有哪些。

1. START TRANSACTION 或者 BEGIN，作用是显式开启一个事务。
2. COMMIT：提交事务。当提交事务后，对数据库的修改是永久性的。
3. ROLLBACK 或者 ROLLBACK TO [SAVEPOINT]，意为回滚事务。意思是撤销正在进行的所有没有提交的修改，或者将事务回滚到某个保存点。
4. SAVEPOINT：在事务中创建保存点，方便后续针对保存点进行回滚。一个事务中可以存在多个保存点。
5. RELEASE SAVEPOINT：删除某个保存点。
6. SET TRANSACTION，设置事务的隔离级别。

需要说明的是，使用事务有两种方式，分别为隐式事务和显式事务。隐式事务实际上就是自动提交，Oracle 默认不自动提交，需要手写 COMMIT 命令，而 MySQL 默认自动提交，当然我们可以配置 MySQL 的参数：

复制代码

```
mysql> set autocommit =0;  // 关闭自动提交
```

复制代码

```
mysql> set autocommit =1;  // 开启自动提交
```

我们看下在 MySQL 的默认状态下，下面这个事务最后的处理结果是什么：

复制代码

```
CREATE TABLE test(name varchar(255), PRIMARY KEY (name)) ENGINE=InnoDB;
BEGIN;
INSERT INTO test SELECT '关羽';
COMMIT;
BEGIN;
INSERT INTO test SELECT '张飞';
INSERT INTO test SELECT '张飞';
ROLLBACK;
SELECT * FROM test;
```

运行结果（1 行数据）：

name
关羽

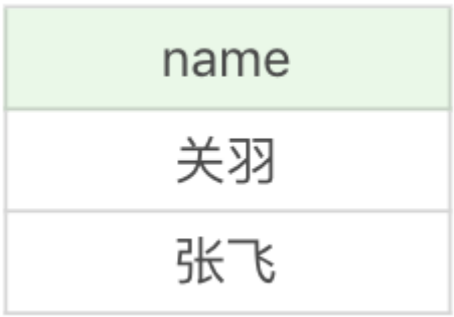
在这个事务中，整个 SQL 一共执行了 2 个事务，第一个是插入“关羽”，提交后执行成功，第二个是插入两次“张飞”，这里需要注意的是，我们将 name 设置为了主键，也就是说主键的值是唯一的，那么第二次插入“张飞”时就会产生错误，然后执行 ROLLBACK 相当于对事务进行了回滚，所以我们看到最终结果只有一行数据，也就是第一个事务执行之后的结果，即“关羽”。

那么如果我们进行下面的操作又会怎样呢?

复制代码

```
CREATE TABLE test(name varchar(255), PRIMARY KEY (name)) ENGINE=InnoDB;
BEGIN;
INSERT INTO test SELECT '关羽';
COMMIT;
INSERT INTO test SELECT '张飞';
INSERT INTO test SELECT '张飞';
ROLLBACK;
SELECT * FROM test;
```

运行结果（2 行数据）：



你能看到这次数据是 2 行，上一次操作我把两次插入“张飞”放到一个事务里，而这次操作它们不在同一个事务里，那么对于 MySQL 来说，默认情况下这实际上就是两个事务，因为在 autocommit=1 的情况下，MySQL 会进行隐式事务，也就是自动提交，因此在进行第一次插入“张飞”后，数据表里就存在了两行数据，而第二次插入“张飞”就会报错：`1062 - Duplicate entry '张飞' for key 'PRIMARY'`。

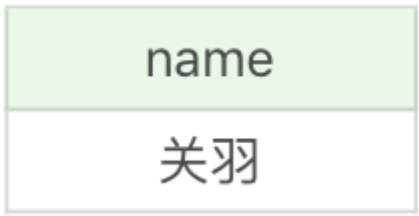
最后我们在执行 ROLLBACK 的时候，实际上事务已经自动提交了，就没法进行回滚了。

同样的我们再来看下这段代码，你又能发现什么不同呢？

复制代码

```
CREATE TABLE test(name varchar(255), PRIMARY KEY (name)) ENGINE=InnoDB;
SET @@completion_type = 1;
BEGIN;
INSERT INTO test SELECT '关羽';
COMMIT;
INSERT INTO test SELECT '张飞';
INSERT INTO test SELECT '张飞';
ROLLBACK;
SELECT * FROM test;
```

运行结果（1 行数据）：



你能看到还是相同的 SQL 代码，只是我在事务开始之前设置了`SET @@completion_type = 1;`，结果就和我们第一次处理的一样，只有一个“关羽”。这是为什么呢？

这里我讲解下 MySQL 中 completion_type 参数的作用，实际上这个参数有 3 种可能：

1. completion=0，这是默认情况。也就是说当我们执行 COMMIT 的时候会提交事务，在执行下一个事务时，还需要我们使用 START TRANSACTION 或者 BEGIN 来开启。
2. completion=1，这种情况下，当我们提交事务后，相当于执行了 COMMIT AND CHAIN，也就是开启一个链式事务，即当我们提交事务之后会开启一个相同隔离级别的事务（隔离级别会在下一节中进行介绍）。
3. completion=2，这种情况下 COMMIT=COMMIT AND RELEASE，也就是当我们提交后，会自动与服务器断开连接。

在上面这段代码里，我使用了 completion=1，也就是说当我提交之后，相当于在下一行写了一个 START TRANSACTION 或 BEGIN。这时两次插入“张飞”会被认为是在同一个事务之内的操作，那么第二次插入“张飞”就会导致事务失败，而回滚也将这次事务进行了撤销，所以你能看到的结果就只有一个“关羽”。

通过这样简单的练习，你应该能体会到事务提交和回滚的操作。

当我们设置 autocommit=0 时，不论是否采用 START TRANSACTION 或者 BEGIN 的方式来开启事务，都需要用 COMMIT 进行提交，让事务生效，使用 ROLLBACK 对事务进行回滚。

当我们设置 autocommit=1 时，每条 SQL 语句都会自动进行提交。
不过这时，如果你采用 START TRANSACTION 或者 BEGIN 的方式来显式地开启事务，那么这个事务只有在 COMMIT 时才会生效，在 ROLLBACK 时才会回滚。

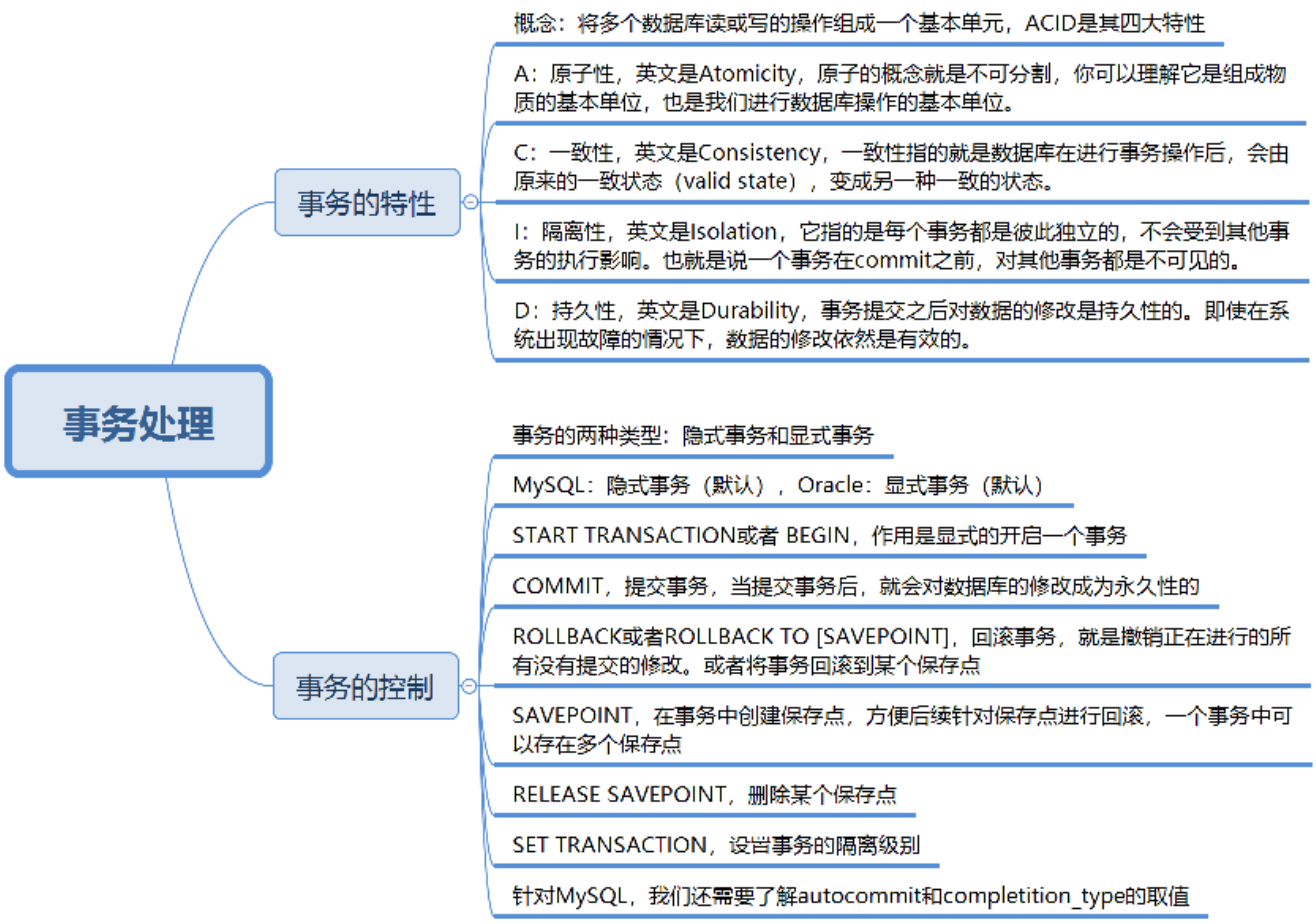
总结

关于 SQL 中的事务处理，内容相对比较多，因此我会采用两节来进行讲解。今天我们对事务的概念进行了理解，并进行了简单的事务操作。我们在做数据库操作的时候，可能会失败，但正是因为有事务的存在，即使在数据库操作失败的情况下，也能保证数据的一致性。同样，多个应用程序访问数据库的时候，事务可以提供隔离，保证事务之间不被干扰。最后，事务一旦提交，结果就会是永久性的，这就意味着，即使系统崩溃了，数据库也可以对数据进行恢复。

在使用事务的过程中，我们会采用控制流语句对事务进行操作，不过在实际操作中，不一定每次使用 COMMIT 或 ROLLBACK 都会成功，你还需要知道当前系统的事务执行方式，也

就是一些常用的参数情况，比如 MySQL 中的 autocommit 和 completion_type 等。

事务是数据库区别于文件系统的重要特性之一，当我们有了事务就会让数据库始终保持一致性，同时我们还能通过事务的机制恢复到某个时间点，这样可以保证已提交到数据库的修改不会因为系统崩溃而丢失。



今天的内容到这里就结束了，你能说一下 MySQL 中都有哪些存储引擎支持事务，通过什么命令可以查看它们吗？另外，你是如何理解事务的特性的？