12 | 事务：怎么确保关联操作正确执行？

朱晓峰 2021-04-03

01:13 1.0x
讲述：朱晓峰 大小：9.46M 时长：10:19

你好，我是朱晓峰。

我们经常会遇到这样的场景：几个相互关联的数据操作，必须是全部执行，或者全部不执行，不可以出现部分执行的情况。比如说，你从微信账号里提现 100 元到银行卡上，这个动作就包括了相互关联的 2 个步骤，首先是微信账号减 100 元，然后是银行卡账号加 100 元（这里假设没有手续费）。假如因为某种异常，这 2 个操作只执行了一个，另外一个没有执行，就会出现你的钱少了 100 元，或者你的钱多了 100 元的情况，这肯定是不能接受的。

如何才能确保多个关联操作全部执行呢？这时就要用到事务了。接下来我就重点讲一讲什么是事务，以及如何正确使用事务。

什么是事务？

事务是 MySQL 的一项功能，它可以使一组数据操作（也叫 DML 操作，是英文 Data Manipulation Language 的缩写，包括 SELECT、INSERT、UPDATE 和 DELETE），要么全部执行，要么全部不执行，不会因为某种异常情况（比如硬件故障、停电、网络中断等）出现只执行一部分操作的情况。

事务的语法结构如下所示：

```
START TRANSACTION 或者 BEGIN （开始事务）
一组DML语句
COMMIT（提交事务）
ROLLBACK（事务回滚）
```

我解释一下这几个关键字。

- **START TRANSACTION 和 BEGIN**：表示开始事务，意思是通知 MySQL，后面的 DML 操作都是当前事务的一部分。
- **COMMIT**：表示提交事务，意思是执行当前事务的全部操作，让数据更改永久有效。
- **ROLLBACK**：表示回滚当前事务的操作，取消对数据的更改。

事务有 4 个主要特征，分别是原子性（atomicity）、一致性（consistency）、持久性（durability）和隔离性（isolation）。

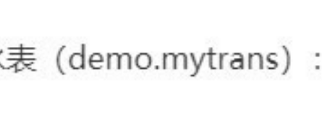
- 原子性：表示事务中的操作要么全部执行，要么全部不执行，像一个整体，不能从中间打断。
- 一致性：表示数据的完整性不会因为事务的执行而受到破坏。
- 隔离性：表示多个事务同时执行的时候，不互相干扰。不同的隔离级别，相互独立的程度不同。
- 持久性：表示事务对数据的修改是永久有效的，不会因为系统故障而失效。

持久性非常好理解，我就不多说了，接下来我重点讲一讲事务的原子性、一致性和隔离性，这是确保关联操作正确执行的关键。

如何确保操作的原子性和数据的一致性？

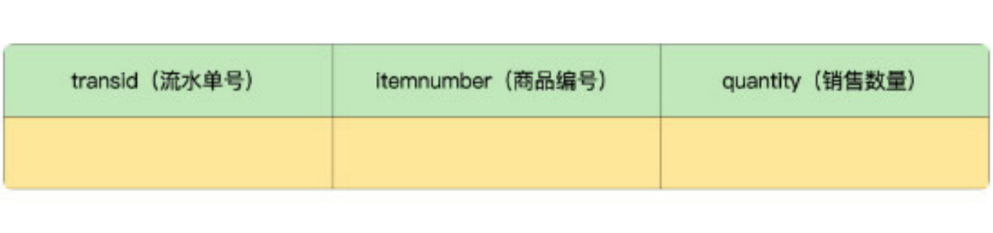
我借助一个超市的收银员帮顾客结账的简单场景来讲解。在系统中，结算的动作主要就是销售流水的产生和库存的消减。这里会涉及销售流水表和库存表，如下所示：

销售流水表（demo.mytrans）：



transid（流水单号）	itemnumber（商品编号）	quantity（销售数量）

库存表（demo.inventory）：



Itemnumber（商品编号）	invquantity（库存数量）
1	10

现在，假设门店销售了 5 个商品编号是 1 的商品，这个动作实际上包括了 2 个相互关联的数据库操作：

1. 向流水表中插入一条“1 号商品卖了 5 个”的销售流水；
2. 把库存表中的 1 号商品的库存减 5。

这里包含了 2 个 DML 操作，为了避免意外事件导致的一个操作执行了而另一个没有执行的情况，我把它们放到一个事务里面，利用事务中数据操作的原子性，来确保数据的一致性。

```
mysql> START TRANSACTION;   -- 开始事务
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO demo.mytrans VALUES (1,1,5); -- 插入流水
Query OK, 1 row affected (0.00 sec)
mysql> UPDATE demo.inventory SET invquantity = invquantity - 5 WHERE itemnumbe
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> COMMIT;                          -- 提交事务
Query OK, 0 rows affected (0.06 sec)
```

然后我们查询一下结果：

```
mysql> SELECT * FROM demo.mytrans;   -- 流水插入成功了
+---------+------------+----------+
| transid | itemnumber | quantity |
+---------+------------+----------+
| 1 | 1 | 5.000 |
+---------+------------+----------+
1 row in set (0.00 sec)
mysql> SELECT * FROM demo.inventory; -- 库存消减成功了
+------------+-------------+
| itemnumber | invquantity |
+------------+-------------+
| 1 | 5.000 |
+------------+-------------+
1 row in set (0.00 sec)
```

这样，通过把 2 个相关操作放到事务里面，我们就实现了一个事务操作。

这里有一个坑，我要提醒你一下。**事务并不会自动帮你处理 SQL 语句执行中的错误**，如果你对事务中的某一步数据操作发生的错误不做处理，继续提交的话，仍然会导致数据不一致。

为了方便你理解，我举个小例子。

假如我们的插入一条销售流水的语句少了一个字段，执行的时候出现错误了，如果我们不对这个错误做回滚处理，继续执行后面的操作，最后提交事务，结果就会出现没有流水但库存消减了的情况：

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO demo.mytrans VALUES (1,5); -- 这个插入语句出错了
ERROR 1136 (21S01): Column count doesn't match value count at row 1

mysql> UPDATE demo.inventory SET invquantity = invquantity - 5 WHERE itemnumbe
Query OK, 1 row affected (0.00 sec)    -- 后面的更新语句仍然执行成功了
Rows matched: 1 Changed: 1 Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.03 sec)   -- 事务提交成功了
```

我们查一下表的内容：

```
mysql> SELECT * FROM demo.mytrans;    -- 流水没有插入成功
Empty set (0.16 sec)
mysql> SELECT * FROM demo.inventory;   -- 库存消减成功了
+------------+-------------+
| itemnumber | invquantity |
+------------+-------------+
| 1 | 5.000 |
+------------+-------------+
1 row in set (0.00 sec)
```

结果显示，流水插入失败了，但是库存更新成功了，这时候没有销售流水，但是库存却被消减了。

这就是因为没有正确使用事务导致的数据不完整问题。那么，如何使用事务，才能避免这种由于事务中的某一步或者几步操作出现错误，而导致数据不完整的情况发生呢？这就要用到事务中错误处理和回滚了：

- 如果发现事务中的某个操作发生错误，要及时使用回滚；
- 只有事务中的所有操作都可以正常执行，才进行提交。

那这里的关键就是判断操作是不是发生了错误。我们可以通过 MySQL 的函数 ROW_COUNT() 的返回，来判断一个 DML 操作是否失败，-1 表示操作失败，否则就表示影响的记录数。

```
mysql> INSERT INTO demo.mytrans VALUES (1,5);
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
| -1 |
+-------------+
1 row in set (0.00 sec)
```

另外一个经常会用到事务的地方是存储过程。由于存储过程中包含很多相互关联的数据操作，所以会大量使用事务。我们可以在 MySQL 的存储过程中，通过获取 SQL 错误，来决定事务是提交还是回滚：

```
mysql> DELIMITER //                    -- 修改分隔符为 //
mysql> CREATE PROCEDURE demo.mytest() -- 创建存储过程
-> BEGIN                               -- 开始程序体
-> DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK; -- 定义SQL操作发生错误是自动回滚
-> START TRANSACTION;                          -- 开始事务
-> INSERT INTO demo.mytrans VALUES (1,5);
-> UPDATE demo.inventory SET invquantity = invquantity - 5;
-> COMMIT;                                     -- 提交事务
-> END
-> //                                          -- 完成创建存储过程
Query OK, 0 rows affected (0.05 sec)

mysql> DELIMITER ;                             -- 恢复分隔符为;
mysql> CALL demo.mytest();                     -- 调用存储过程
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM demo.mytrans;             -- 销售流水没有插入
Empty set (0.00 sec)
mysql> SELECT * FROM demo.inventory;           -- 库存也没有消减，说明事务回滚了
+------------+-------------+
| itemnumber | invquantity |
+------------+-------------+
| 1 | 10.000 |
+------------+-------------+
1 row in set (0.00 sec)
```

这里，我们要先通过“DELIMITER //”语句把 MySQL 语句的结束标识改为“//”（默认语句的结束标识是“;”）。这样做的目的是告诉 MySQL 一直到“//”才是语句的结束，否则，MySQL 会在遇到第一个“;”的时候认为语句已经结束，并且执行。这样就会报错，自然也就没办法创建存储过程了。

创建结束以后，我们还要录入“//”，告诉 MySQL 存储过程创建完成了，并且通过“DELIMITER ;”，再把语句结束标识改回到“;”。

关于存储过程，我会在后面的课程里给你详细介绍。这里你只需要知道，在这个存储过程中，我使用了“DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;”这个语句，来监控 SQL 语句的执行结果，一旦发发生错误，就自动回滚并退出。通过这个机制，我们就实现了对事务中的 SQL 操作进行监控，如果发现事务中的任何 SQL 操作发生错误，就自动回滚。

总之，**我们要把重要的关联操作放在事务中，确保操作的原子性，并且对失败的操作进行回滚处理**。只有这样，才能真正发挥事务的作用，保证关联操作全部成功或全部失败，最终确保数据的一致性。

如何用好事务的隔离性？

接下来，我们再学习下如何用好事务的隔离性。

超市经营者提出，门店要支持网上会员销售，现在我们假设会员张三是储值会员，他的会员卡里有 100 元。张三用会员卡到门店消费 100 元，他爱人用他的会员卡在网上消费 100 元。

张三在门店消费结算的时候，开启了一个事务 A，包括这样 3 个操作：

1. 读取卡内金额为 100；
2. 更新卡内金额为 0；
3. 插入一条销售流水。

张三的爱人在网上购物，开启了一个事务 B，也来读取卡内金额。如果 B 读取卡内金额的操作，发生在 A 更新卡内金额之后，并且在插入销售流水之前，那么 B 读出的金额应该是多少呢？如果 B 读出 0 元，那么，A 有可能由于后面的操作失败而回滚。因此，B 可能会读到一条错误信息，而导致本来可以成功的交易失败。有什么办法可以解决这个问题呢？

这个时候，就会用到 MySQL 的另外一种机制：“锁”。MySQL 可以把 A 中被修改过而且还没有提交的数据锁住，让 B 处于等待状态，一直到 A 提交完成，或者失败回滚，再释放锁，允许 B 读取这个数据。这样就可以防止因为 A 回滚而导致 B 读取错误的可能了。

MySQL 中的锁有很多种，功能也十分强大。咱们这门课里不要求你掌握锁，你只要知道，MySQL 可以用锁来控制事务对数据的操作，就可以了。

通过对锁的使用，可以实现事务之间的相互隔离。**锁的使用方式不同，隔离的程度也不同。**

MySQL 支持 4 种事务隔离等级。

1. READ UNCOMMITTED：可以读取事务中还未提交的被更改的数据。
2. READ COMMITTED：只能读取事务中已经提交的被更改的数据。
3. REPEATABLE READ：表示一个事务中，对一个数据读取的值，永远跟第一次读取的值一致，不受其他事务中数据操作的影响。这也是 MySQL 的默认选项。
4. SERIALIZABLE：表示任何一个事务，一旦对某一个数据进行了任何操作，那么，一直到这个事务结束，MySQL 都会把这个数据锁住，禁止其他事务对这个数据进行任何操作。

一般来讲，使用 MySQL 默认的隔离等级 REPEATABLE READ，就已经够了。不过，也不排除需要对一些关键的数据操作，使用最高的隔离等级 SERIALIZABLE。

举个例子，在我们的超市项目中，就对每天的日结操作设置了最高的隔离等级。因为日结要进行大量的核心数据计算，包括成本、毛利、毛利率、周转率，等等，并把结果保存起来，作为各类查询、报表系统、决策支持模块的基础，绝对不能出现数据错误。

当然，**计算完成之后，你也不要忘记把隔离等级恢复到系统默认的状态**，否则，会对日常的系统营运效率产生比较大的影响。

事务的隔离性对并发操作非常有用。当许多用户同时操作数据库的时候，隔离性可以确保各个连接之间互相不影响。这里我要提醒你的是，正确设置事务的隔离等级很重要。

一方面，**对于一些核心的数据更改操作，你可能需要较高的隔离等级**，比如涉及金额的修改；另一方面，**你要考虑资源的消耗，不能使系统整体的效率受到太大的影响**。所以，要根据具体的应用场景，正确地使用事务。

总结

事务可以确保事务中的一系列操作全部被执行，不会被打断；或者全部不被执行，等待再次执行。事务中的操作，具有原子性、一致性、永久性和隔离性的特征。但是这并不意味着，被事务包裹起来的一系列 DML 数据操作就一定会全部成功，或者全部失败。你需要对操作是否成功的结果进行判断，并通知 MySQL 针对不同情况，分别完成事务提交或者回滚操作，才能最终确保事务中的操作全部成功或全部失败。

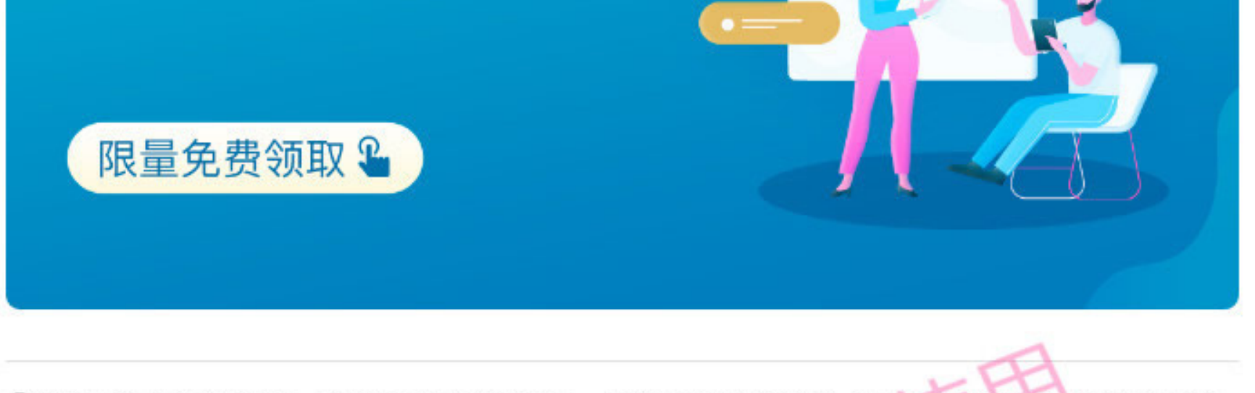
MySQL 支持 4 种不同的事务隔离等级，等级越高，消耗的系统资源也越多，你要根据实际情况进行设定。

在 MySQL 中，并不是所有的操作都可以回滚。比如创建数据库、创建数据表、删除数据库、删除数据表等，这些操作是不可以回滚的，所以，你在操作的时候要特别小心，特别是在删除数据库、数据表时，最好先做备份，防止误操作。

思考题

学完了这节课以后，如果现在有人对你说，事务就是确保事务中的数据操作，要么全部正确执行，要么全部失败，你觉得这句话对吗？为什么？

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，也欢迎你把它分享给你的朋友或同事，我们下节课见。



精选留言(10)

朱晓峰
置顶

你好，我是朱晓峰，下面我就来公布一下上节课思考题的答案：

上节课，我们学习了什么是索引、如何创建和使用索引。下面是思考题的答案：

我会用门店编号、销售日期、商品编号、销售金额和毛利这些字段，分别创建索引，理由是这些字段经常会被用作筛选条件来进行查询。

2021-04-21

Harry

学了这节课，还真不敢说自己知道什么是事务！

2021-04-04 6

Devo

“数据操作，要么全部正确执行，要么全部失败”，我认为不对。数据操作的正确性依赖于事务隔离级别，如果在rc下，那么事务A的执行过程中就有可能读取到其它事务的提交结果，从而导致数据计算错乱，所以在数据操作的场景下务必要根据业务场景设计好事务隔离级别，避免数据计算错乱，望老师指正。

2021-04-07 1

EatheinWong

或者可以这么说：事务保证让一系列动作作为一个整体成功得到执行，如果执行结果全部符合预期则视为执行成功，那么就提交。如果执行不是全部成功，那么就撤回所有操作。

2021-04-12

EatheinWong

看了老师的讲解，我觉得对事务简单的理解就是：让一系列动作先后执行，如果执行结果符合预期(没有异常)，那么一起提交，坐实；如果结果不符合预期(异常发生)，那么就撤回所有动作。

2021-04-12

青生先森

老师，什么情况下设置为可重复读，读已提交呢？

2021-04-06 3

molingwen

是否能提供一下这个测试数据库，非常感谢

2021-04-05

洛奇

文中，张三和张三爱人消费会员卡购物的例子里，正确的事务隔离级别应该是SERIALIZABLE吗？其他的三个隔离级别，我认为都存在发生错误的可能性。比如，默认的可重复读的隔离级别下，两个事务一起并行执行时，都只会取到100元，两个事务提交后，会员卡余额变成0，但消费了两次！

2021-04-05 2

Harry

学习了要监控事务中失败的 SQL 操作并对其进行回滚处理。

但对于如何用好事务的隔离性，心里没底。主要是不知道如何使用锁来操作不同的隔离等级。

2021-04-04

lesserror

每一讲都有收获。

课后思考题的答案已经在总结的第一段话中给出了。

关于MVCC和锁是面试的常见问题。本专栏侧重入门。这部分内容看《MySQL实战45讲》和书籍《MySQL是怎样运行的》。

以上资料都认真看了，基本应对数据库的问题可以做到胸有成竹了！

2021-04-03