**[MySQL ------ 事务处理（transaction）（二十八）](https://www.cnblogs.com/obge/p/13149883.html)**

MySQL支持的几种数据库引擎，并非所有的引擎都支持明确的事务处理管理，MYISAM 和 INNODB 是最常用的引擎，而MYISAM 不支持事务处理，**innodb支持事务处理**，所以一定要正确 使用引擎类型。

**事务处理（transaction processing）：可以用来维护数据库的完整性，保证成批的MySQL操作要么完全执行，要么完全不执行。**

**事务（transaction）: 指将一系列数据操作捆绑成一个整体进行统一管理。如果某一事务执行成功，则在该事务中进行的所有数据更改都会提交，成为数据库中永久组成部分。如果事务执行时遇到错误且必须取消或回滚，则所有数据将全部恢复到操作前的状态，所有数据的更改均被清除。**

在银行业务中有一个记账原则，有借有贷，借贷相等，大致意思就是，当一笔转账业务在进行时，两方之间的总额在转账前后还是一致的。因此就需要事务来进行管理，不能我这边减少了，中间发生错误对方没收到，帐就这么对不上了。

 事务是一种机制、一个操作序列，包含了一组数据库操作命令，并且把所有的命令作为一个整体一起向系统提交或撤销操作请求，要么全都执行成功， 要么全都不执行。因此事务是一个不可分割的工作逻辑单元。

**在数据库系统上执行并发操作时，事务是作为最小的控制单元来进行使用，特别适用于多用户同时操作的数据库系统**。如：订票、银行、证券交易等系统

**一、四大特性**

**事务作为一个逻辑工作单元，主要有四个属性：原子性（atomicity）、一致性（consistency）、隔离性（isolation）和持久性（durability）通常放到一块简称为ACID**

(1)、原子性

事务是一个完整的操作，事务的个元素是不可分的（原子的）。事务中的所有元素必须作为一个整体提交或回滚，如果事务中的任何元素失败，则整个事务都将失败。

如：A 给 B 转账，A若转账成功，B 就收到，否则，转账失败

(2)、一致性

当事务完成时，数据必须处于一致状态。也就是说在事务开始之前数据库中存储的数据处于一致状态。在正在进行的事务中，数据可能处于不一致状态（如：数据被修改），但当事务完成时，数据必须再次回到已知的一致状态。通过事务对数据所做的修改不能损坏数据，或者说事务不能使数据存储处于不稳定的状态。

如： A 给 B 转账，转账前两者总额处于一致状态，在进行转账时若A 修改了，B也修改了，此时二者总额与之前一致，则处于一致状态，若A改了，B未改，二者总额与之前不一致则处于不一致状态。

(3)、隔离性

 对数据进行修改的所有并发事务是彼此隔离的，这表明事务是必须独立的，它不应以任何方式依赖于或影响其他事务。修改数据的事务可以在另一个使用相同数据的事务开始之前访问这些数据，或者在另一个使用相同数据的事务结束之后访问这些数据，另外，当事务修改数据时，如果任何其他进程正在同时使用相同的数据，则直到该事务成功提交之后，对数据的修改才能生效。

如：A与B之间转账，C与D之间转账，永远是相互独立。

(4)、持久性

指不管系统是否发生了故障，事务处理的结果都是永久的。

一个事务成功执行完成之后，它对于数据库的改变是永久性的，即使系统出现故障也是如此，也就是一旦事务被提交，事务的效果会被永久地保留在数据库中。

**二、执行事务**

任何一种数据库都会拥有各种各样地日志，用来记录数据地运行情况、日常操作、错误信息等，MySQL也不列外。如：当用户root 登录到MySQL服务器，就会在日志文件里记录该用户的登录时间、执行操作等，为了维护Mysql服务器经常需要在MySQL数据库中进行日志操作

**MySQL中支持事务地存储引擎有InnoDB 和 BDB**，**InnoDB存储引擎管理事务主要通过UNDO日志和REDO 日志实现**。

**UNDO日志**：复制事务执行前的数据，用于在事务发生异常时回滚数据。

**REDO日志**：记录在事务执行中，对数据进行的每条更新操作，当事务提交时，该内容将被刷新到磁盘。

默认设置下，每条SQL就是一个事务，即执行SQL语句后自动提交。为了达到将几个操作作为一个整体的目的，需要使用 BEGIN 和 START TRANSACTION 开启一个事务，或执行命令 SET AUTOCOMMIT = 0,来禁止当前会话的自动提交，命令后面的语句作为事务的开始。

**One、在使用事务和事务处理时常用的关键字**

**事务**（**transaction**）指一组SQL语句

**回退**（**rollback**）指撤销指定SQL语句的过程

**提交**（**commit**）指将未存储的SQL语句结果写入数据表

**保留点**（**savepoint**）指事务处理中设置的临时占位符（placeholder），你可以对它发布回退（与回退整个事务处理不同）

**注意：当commit 或 rollback 语句执行后，事务会自动关闭**

**Two、开启事务**

管理事务处理的关键在于将sql语句组分解成逻辑块，并确定规定数据何时应该回退，何时不应该回退。

-- 标识事务的开始

start transaction

或

begin

这个语句显示地标记一个事务地起始点。

**Three、使用 ROLLBACK（回退：用来管理insert 、update、delete 语句）**

**ROLLBACK 只能在一个事务处理内使用，也就是在执行了一条start transaction之后，将start transaction 之后的所有语句回退。**

[复制代码](javascript:void(0);)

-- 开启事务

start transaction;

-- 在事务中删除 ordertotals

delete from ordertotals;

-- 查看一下还有没

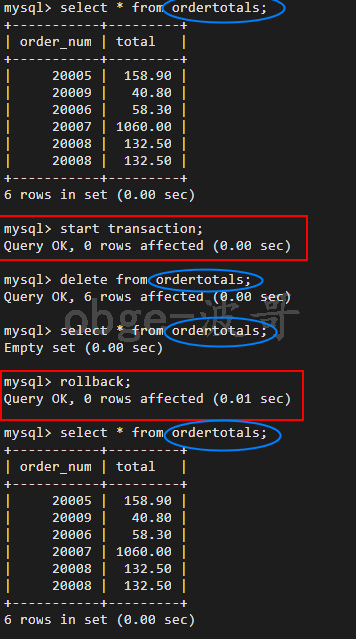
select \* from ordertotals;

-- 回退，反悔了不能删除

rollback;

[复制代码](javascript:void(0);)

rollback 清楚自事务起始点至该语句所做地所有数据更新操作，将数据状态回滚到事务开始前，并释放由事务控制地资源。



 上述，首先查询不为空的一个表，然后开启了一个事务处理，用一条delete语句删除这个表中的所有行，验证一下是否空了，这时用一条回退（rollback）语句，将start transaction 之后的所有语句回退，最后验证一下是否成功。

注意：

**事务处理ROLLBACK用来管理insert 、update、delete 语句**，不能回退Crete、drop、select操作（虽然可以执行但是没用）。

**Four、使用COMMIT（提交）**

 一般的MySQL语句都是**直接针对数据表执行和编写的**。这就是所谓的**隐含提交（implicit commit）,即提交（也可以叫写、保存）操作是自动进行的。**

注意：在事务处理块中，提交不会隐含地进行，为了进行明确的提交，要使用commit 语句。

[复制代码](javascript:void(0);)

-- 从数据库中删除订单号为 20007的订单，注意要先删除明细在删除订单,不然由于主外键的关系导致删除不了

-- 开启事务

start transaction;

-- 删除明细表中

delete from orderitems where order\_num = **20007**;

-- 删除订单表中

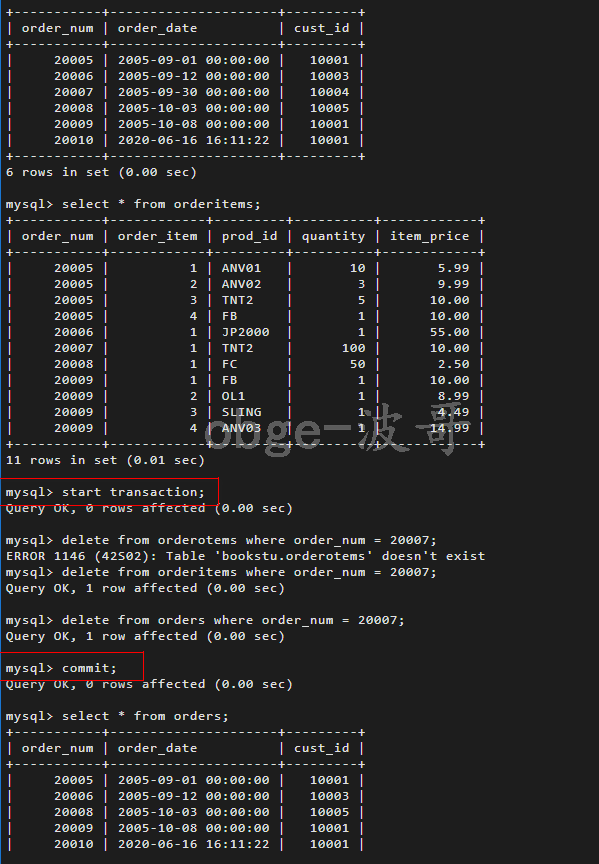
delete from orders where order\_num = **20007**;

-- 提交

commit;

[复制代码](javascript:void(0);)

commit 标志一个事务成功提交，自事务开始至提交语句之间执行的所有数据更新将永久地保存在数据库文件中，并释放连接时占用地资源。



 上述中，从数据库中完全删除订单20010.，主要涉及更新两个数据表orders 和 orderitems，所以使用事务处理块来保证订单不被部分删除，最后使用commit 语句仅在不出错时写出更改，如果第一条删除语句起作用，但第二条失败，则delete语句不会被提交（而会被自动撤销）

**Five、使用保留点（savepoint）**

简单的rollback 和commit 语句就可以写入或撤销整个事务处理，但是更复杂事务处理可能会部分提交或回退，所以引入了保留点

如之前的添加订单场景，如果发生错误回退到 添加 orders 行之前就行，不需要回退到客户表

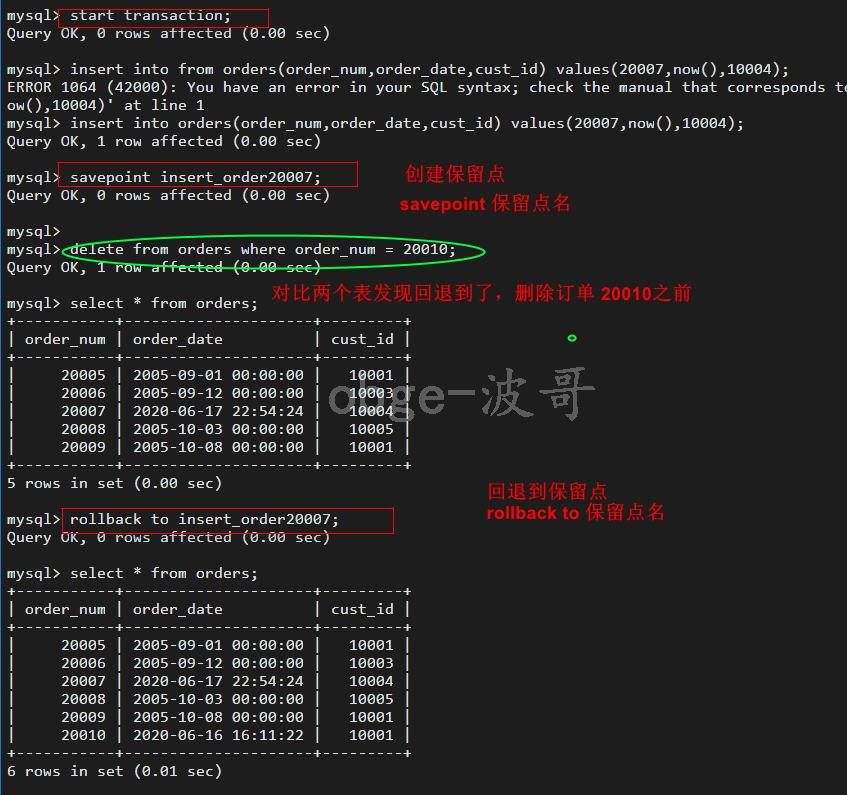
为了支持回退部分事务处理，必须能在事务处理块中合适的位置防止占位符，这样如果需要回退，就可以回退到某个占位符。而这些占位符就称为保留点(创建时可以使用 savepoint 声明)

-- 创建保留点

savepoint 保留点名

-- 回退到保留点

rollback to 保留点名



注意：

1、每个保留点的名字必须要唯一而且要有意义，以便在回退时，MySQL知道回到哪

2、可以设置任意多的保留点，而且时越多越好，因为保留点越多，就越可以灵活的进行回退

3、保留点在事务处理完成（即执行一条Rollback或commit ）之后自动释放，在MySQL 5 后也可以用RELEASE SAVEPOINT 明确的释放保留点。

**Six、更改默认的提交 (autocommit)**

默认的MySQL行为时自动提交所有更改，也就是任何时候执行一条MySQL语句，该语句对表的更改时立即执行生效的。如果要想关闭可以使用以下语句；

-- 关闭默认提交

set autocommit = **0**;

注意：

1、autocommit 决定是否自动提交更改，不管有没有commit 语句，设置autocommit 为 0（假）指示MySQL不自动提交更改，直到autocommit设置为真时为止。

2、autocommit 标志是针对每个连接而不是服务器。

**三、编辑事务时要遵守以下原则**

**(1)、事务尽可能简短。**

事务启动至结束在数据库管理系统中会保留大量资源，以保证事务地原子性、一致性、隔离性和持久性。如果在多用户系统中，较大地事务将会占用系统大量地资源，使系统不堪重负、会影响软件地运行性能，甚至导致系统崩溃。

**(2)、事务中访问地数据量尽量最少。**

当并发执行事务处理时，事务操作地数据量越少，事务之间对操作数据地争夺就越少。

**(3)、查询数据时不要使用事务。**

对数据进行浏览查询操作并不会更新数据库地数据，因此不使用事务查询数据，以避免占用过量地系统资源。

**(4)、在事务处理过程中尽量不要出现等待用户输入地操作。**

在处理事务地过程中，如果需要等待用户输入数据，那么事务会长时间地占用资源，有可能造成系统阻塞。

end!!!!!

在关系型数据库设计把数据存储在多个表中，使数据更容易操纵、维护和重用。

如：订单存储在orders 表和orderitems  表中，order存储实际的订单，而orderitems 存储订购的各项物品，这两个表使用主键相互关联，又包含客户和产品信息的其他表相关联

给系统添加订单的过程如下

1、检查数据库中是否存在相应的客户（从customer 表中查询），如果不存在添加他/她

2、检索客户的ID

3、添加一行到orders 表，把它与客户ID关联

4、检索orders表中赋予的新的订单ID

5、对于订购的每个物品在orderitems 表中添加一行，通过检索出来的ID 把它与orders表关联（以及通过产品ID 与product表关联）

从中可以看出，一个添加订单的过程涉及的客户表，订单表，订单明细表，产品表（要是复杂一点可能还涉及到物流配送，库存占用）

**如果要是数据库发生了某种故障（如超出磁盘空间，安全限制，表锁等）阻止了这个过程的完成。**

如果故障发生在添加了客户之后，orders 表添加之前，不会有什么问题。某些客户没有订单是完全合法的。在重新执行此过程时，所插入的客户将被检索和使用，可以有效地从出故障的地方开始执行此过程。

如果故障发生在order行添加之后，orderitems行添加之前，会导致数据库中有一个空的订单

如果故障发生在系统在给orderitems添加行时，会导致数据库出现一个不完整订单而且你还不知道。

如何解决，这个时候事务处理就发挥作用了，**事务处理是一种机制，用来管理必须成批的执行MySQL操作，以保证数据库不包含不完整的操作结果。**

利用事务机制可以保证一组操作不会中途停止，他们要么作为整体执行，要么完全不执行（除非明确指示），如果没有发生错误，整组sql语句都会执行将数据写入到数据库，如果出现错误，则进行回退（撤销）恢复到这组操作之前的数据状态。

**使用事务后相同的操作就会变成**

1、检查数据库中是否存在相应的客户，如果不存在，添加他/她

2、提交客户信息

3、检索客户的ID

4、添加一行到orders 表

5、如果在添加行到orders 表时处理故障，回退

6、检索orders 表中赋予的新订单ID

7、对于订购的每项物品，添加新行到orderitems表

8、如果在添加新行到orderitems时出现故障，回退所添加的orderitems 行和orders 行

9、提交订单信息

分类: [MySQL](https://www.cnblogs.com/obge/category/1654522.html)