案例：银行的数据库里面存储着用户的账户信息表，当用户 A 想用户 B 转账的时候，正常情况下，A 账户的余额减少，B 账户的余额增加；但是由于某种原因（例如突然断电），当 A 账户的余额减少之后，B 账户的余额并没有增加，这就造成了数据库数据的安全隐患。

解决方案：当 A 账户的余额减少之后，不要立即修改数据表，而是在确认 B 账户的余额增加之后，同时修改数据表。

事务

通过前面的案例及解决方案，我们就引出了一个全新的概念，那就是：事务，即

一系列将要发生或正在发生的连续操作。

而事务安全，是一种保护连续操作同时实现（完成）的机制。事务安全的意义就是，保证数据操作的完整性。

首先，执行如下 SQL 语句，创建银行账户表并插入数据：

-- 创建银行账户表

create table bank\_account(

id int primary key auto\_increment,

cardno varchar(16) not null unique comment 'bank card number',

name varchar(20) not null,

money decimal(10,2) default 0.0 comment 'account balance'

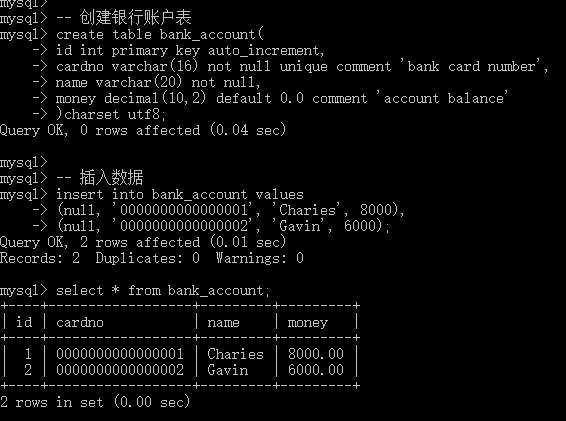
)charset utf8;

-- 插入数据

insert into bank\_account values

(null, '0000000000000001', 'Charies', 8000),

(null, '0000000000000002', 'Gavin', 6000);

[](https://camo.githubusercontent.com/e37aef68455292dc4d67ae2982a4fa7d81f28c4f/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231313935333030353831)

接下来，让我们一起了解事务的操作。

事务操作

事务操作，分为两种：自动事务（默认的），手动事务。

在这里，以银行账户的余额增减为例，我们来了解手动事务的操作流程。

第 1 步：开启事务，告诉系统以下所有操作，不要直接写入数据库，先存到事务日志。

基本语法：start transaction;

执行如上 SQL 语句，开启事务：

-- 开启事务

start transaction;

[](https://camo.githubusercontent.com/290f22d14f57d78887f90a93125758c5d78f051d/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231323030343236383731)

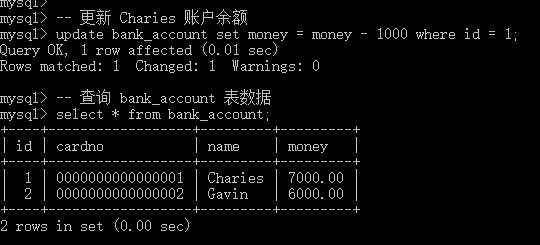
第 2 步：减少 Charies 账户的余额

-- 更新 Charies 账户余额

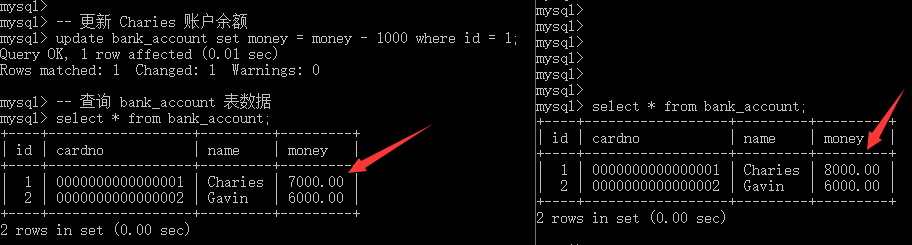
update bank\_account set money = money - 1000 where id = 1;

-- 查询 bank\_account 表数据

select \* from bank\_account;

[](https://camo.githubusercontent.com/35d1bda1f423b15f3022d97ad1bb336603290aaa/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231323031303434313532)

如上图所示，Charies 账户的余额显示减少1000，但实际上，由于我们开启了事务，数据表真实的数据，并没有同步更新。为了验证这个论断，我们重新打开一个数据库客户端，查询bank\_account表的数据：

[](https://camo.githubusercontent.com/c48901eec04f53c0a603d81735d55f520c38eba0/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231323031353430373635)

如上图所示，显然数据库的事务安全机制起了作用，当我们开启（手动）事务之后，其后一系列操作并没有直接写入数据库，而是存入了事务日志。在这里，我们并没有打开数据库事务的日志进行验证，因为事务日志存储的是经过编译之后的字节码文件。

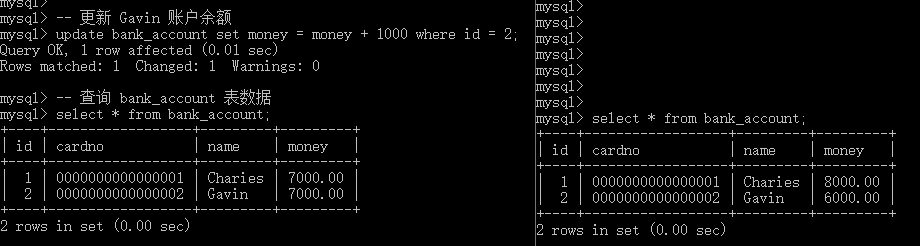
第 3 步：增加 Gavin 账户的余额

-- 更新 Gavin 账户余额

update bank\_account set money = money + 1000 where id = 2;

-- 查询 bank\_account 表数据

select \* from bank\_account;

[](https://camo.githubusercontent.com/79c593fb9ce44cec067254c7f8b476158f24d6ba/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231323032333331303934)

如上图所示，Gavin 账户的余额显示增加1000，但是，由于我们开启了事务，数据表真实的数据，仍然没有同步更新。

第 4 步：提交事务或回滚事务

提交事务基本语法：commit;

回滚事务基本语法：rollback;

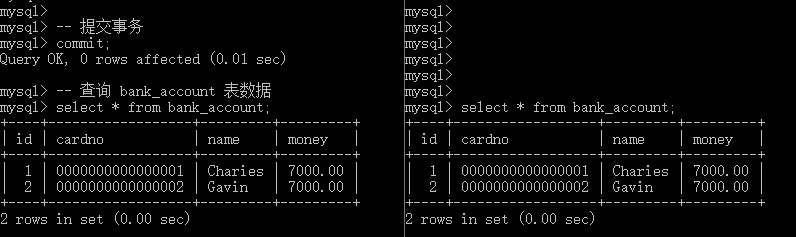
如果我们选择提交事务，则将事务日志存储的记录直接更新到数据库，并清除事务日志；如果我们选择回滚事务，则直接将事务日志清除，所有在开启事务至回滚事务之间的操作失效，保持原有的数据库记录不变。在这里，我们以提交事务为例：

-- 提交事务

commit;

-- 查询 bank\_account 表数据

select \* from bank\_account;

[](https://camo.githubusercontent.com/44cbf6fcced3c6e4f02a017661e9095ae2105c9e/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303231323033343033393535)

如上图所示，当我们提交事务之后，数据库的真实记录更新，两个客户端的数据一致。

在此，值得我们注意的是：当我们提交事务之后，在进行回滚事务是不起作用的，因为事务日志在提交事务的同时已经被清除啦！

此外，我们还要知道：现阶段，只有 InnoDB 和 BDB 两个存储引擎是支持事务安全机制的，其中 InnoDB 免费，BDB 收费。因此，InnoDB 使用的最为广泛。