

# 【MySQL】事务(五)

🚗 MySQL学习·第五站~

📌 本文已收录至专栏: [MySQL通关路](#)

❤️ 文末附全文思维导图, 感谢各位点赞收藏支持~

★ 学习汇总贴, 超详细思维导图: [【MySQL】学习汇总\(完整思维导图\)](#)

## 一.引入

事务, 指的是一组操作的集合, 它是一个不可分割的工作单位, 它会把这个集合中所有的操作作为一个整体一起向系统提交或撤销操作请求, 即这些操作要么同时成功, 要么同时失败。

事务常常用于在需要操作多条记录或多张表的情况下, 为了避免在执行过程中出现异常行为导致数据一致性被破坏, 这时候我们就需要开启事务。

就比如最经典的银行转账问题:

- 张三给李四转账1000块钱, 张三银行账户的余额减少1000, 而李四银行账户的余额要增加 1000。这一组操作就必须在一个事务的范围内, 即要么都成功, 要么都失败。总不可能在出现异常后, 张三的余额减少了, 李四的余额却没有增加?

假设我们没有开启事务, 也就是这一系列操作不在一个事务的范围内:

- 这是我们的理想情况

**无异常, 转账都成功!**

查询张三账户余额

张三账户余额-1000

`update account set money = money - 1000 where name = '张三';`

李四账户余额+1000

`update account set money = money + 1000 where name = '李四';`

id	name	money
1	张三	1000
2	李四	3000

CSDN @观止study

- 假设抛出了异常

查询张三账户余额

张三账户余额-1000

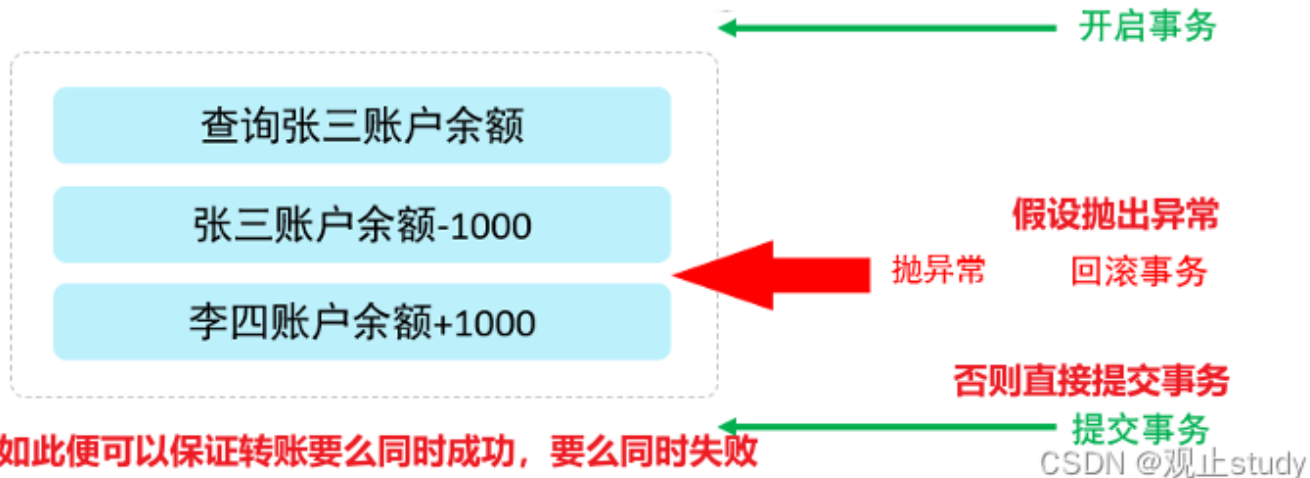
李四账户余额+1000

假设张三转账后  
抛异常

id	name	money
1	张三	1000
2	李四	2000

我们可能看到, 张三的余额减少了1000, 但是李四的余额却并没有增加!!! 这显示不是我们所期望看到的 CSDN @观止study

- 为了解决上述的问题, 我们就需要通过开启事务来完成, 只需要在业务逻辑执行之前开启事务, 执行完毕后必须提交事务, 如果执行过程中报错, 则回滚事务, 把数据恢复到事务开始之前的状态。



值得一提的是，默认MySQL的事务是自动提交的，也就是说，当执行完一条DML语句时，MySQL会立即隐式的提交事务。即默认情况下，一条SQL语句就是一个事务。假如关闭自动提交则必须在每次执行SQL之后手动提交事务，否则SQL不生效。

## 二.事务操作

### (1) 查看事务提交方式

- 语法

```
SELECT @@autocommit;
```

```
1 SELECT @@autocommit;
```

信息	结果1	概况	状态
@@autocommit	1		

CSDN @观止study

其中 1 表示自动提交事务，0 表示手动提交事务。

### (2) 设置事务提交方式

- 语法

```
SET @@autocommit = 0; # 设置为手动提交事务
SET @@autocommit = 1; # 设置为自动提交事务
```

```
2
3 # 设置为手动提交
4 SET @@autocommit = 0;
5 # 查看事务提交方式
6 SELECT @@autocommit;
```

信息	结果1	概况	状态
@@autocommit	0		

CSDN @观止study

### (3) 提交事务

- 语法

```
COMMIT;
```

- 上述我们把事务的提交方式修改为了手动提交，接下来我们来验证一下“**关闭自动提交事务则必须在每次执行SQL之后手动提交事务，否则SQL不生效**”。

id	name	money
1	张三	2000
2	李四	2000

但是数据库内数据并未增加

```
8 update account set money = money + 1000 where name = '张三';
```

信息	概况	状态
[SQL]update account set money = money + 1000 where name = '张三';		
受影响的行: 1		
时间: 0.001s		

窗口显示执行成功

CSDN @观止study

我们关闭了自动提交事务，执行SQL后发现执行成功，但没有提交事务，查看数据库发现数据并未发生变化。

id	name	money
1	张三	3000
2	李四	2000

再次查看数据库，发现数据这才更新

```
10 commit;
```

信息	概况	状态
[SQL]commit;		
受影响的行: 0		
时间: 0.002s		

提交事务后

CSDN @观止study

待我们提交事务后，数据才更新！

### (4) 回滚事务

- 语法

```
ROLLBACK;
```

- 当我们执行事务的过程中碰到了异常导致中止，可以使用 `rollback` 使得事务回滚，即恢复事务开始执行之前的状态。

### (5) 开启事务

- 语法

```
START TRANSACTION; 或 BEGIN;
```

- 我们除了可以以关闭自动提交事务，然后手动commit的方式控制事务以外，还可以通过在SQL开始执行之前，先执行 `START TRANSACTION; 或 BEGIN;` 然后执行SQL，最后commit的方式控制事务。

id	name	money
1	张三	3000
2	李四	1000

## 执行转账操作

```
# 开启事务
START TRANSACTION;
# 执行SQL
update account set money = money + 1000 where name = '张三';
update account set money = money - 1000 where name = '李四';
# 提交事务
commit;
```

CSDN @观止study

## 三.四大特性

事务有着四大特性（ACID）：

- **原子性**（Atomicity）：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。



同时成功

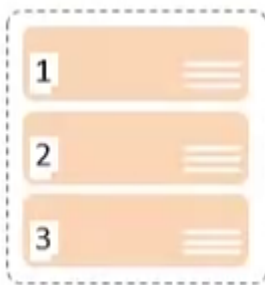
或



同时失败

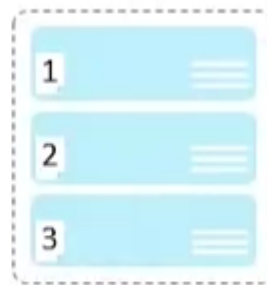
CSDN @观止study

- **一致性**（Consistency）：事务完成时，必须使所有的数据都保持一致状态。
- **隔离性**（Isolation）：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环境下运行。



事务A

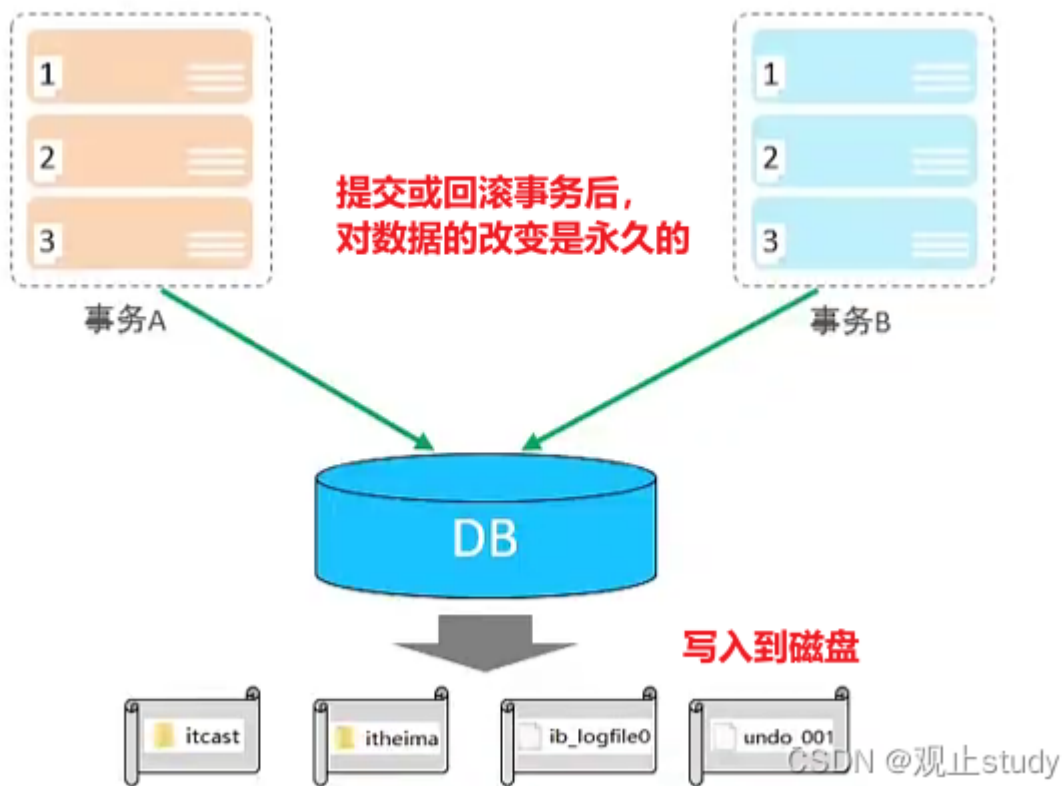
独立运行



事务B

CSDN @观止study

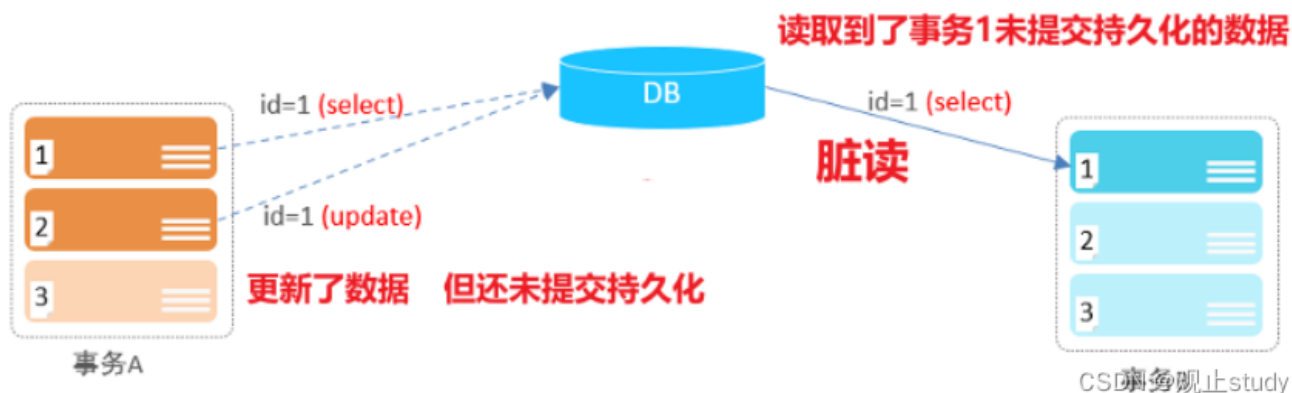
- **持久性**（Durability）：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。



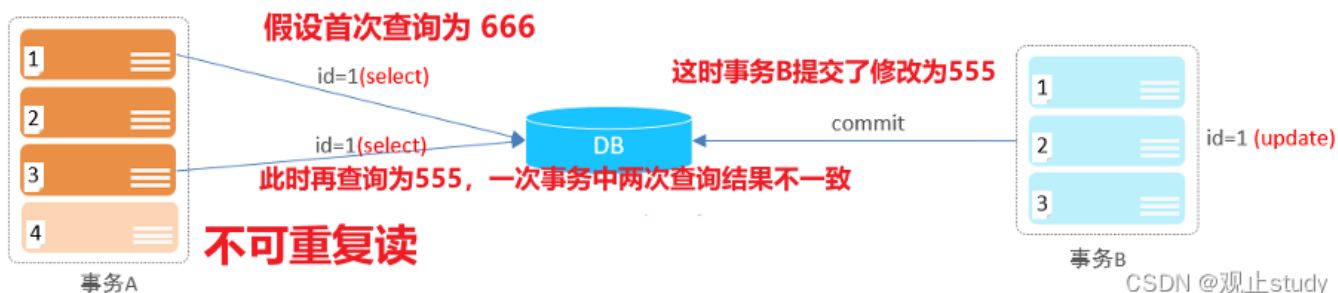
## 四.并发事务问题

多个事务同时操作某一个数据库或者某一张表时，可能会产生一系列问题：

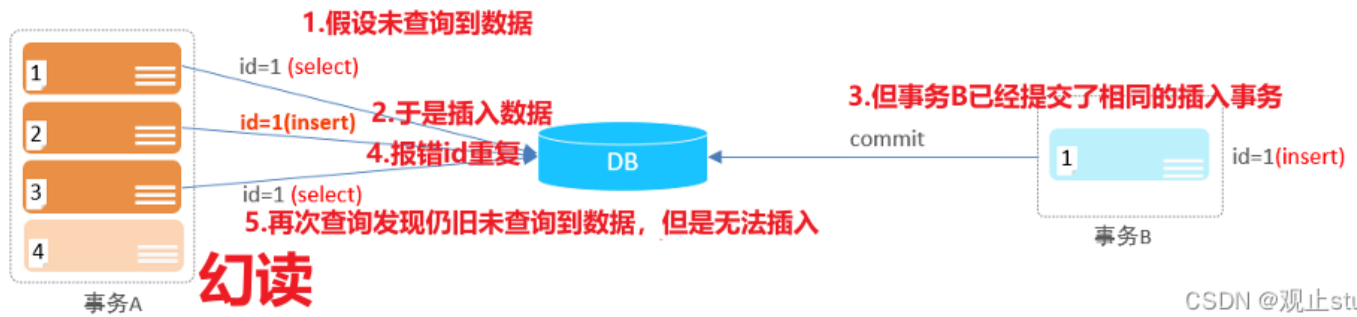
- **脏读**：一个事务读到另外一个事务还没有提交的数据



- **不可重复读**：一个事务先后读取同一条记录，但两次读取的数据不同，称之为不可重复读。



- **幻读**：一个事务按照条件查询数据时，没有对应的数据行，但是在插入数据时，又发现这行数据已经存在，好像出现了"幻影"。



## 五.事务的隔离级别

为了解决并发事务所引发的问题，在数据库中引入了事务隔离级别。主要有以下几种：

隔离级别（发生-√，不发生-×）	脏读	不可重复读	幻读
Read uncommitted（读未提交）	√	√	√
Read committed（读已提交）	×	√	√
Repeatable Read（可重复读）（默认）	×	×	√
Serializable（串行化）	×	×	×

MySQL默认的隔离级别为 Repeatable Read，也就是只会产生幻读问题。当然，随着安全系数的增加，数据库的性能也有所下降。

- 查看事务隔离级别

```
SELECT @@TRANSACTION_ISOLATION;
```

1 SELECT @@TRANSACTION\_ISOLATION;|

### MySQL默认隔离级别

信息	结果1	概况	状态
	@@TRANSACTION_ISOLATION		
	REPEATABLE-READ		

CSDN @观止study

- 设置事务隔离级别

```
SET [ SESSION | GLOBAL ] TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

3 SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE

### 设置当前会话隔离级别为串行化

信息	结果1	概况	状态
	@@TRANSACTION_ISOLATION		
	SERIALIZABLE		

CSDN @观止study

## 七.全文概览

事务是一组操作的集合，它是一个不可分割的工作单位，事务会把所有的操作作为一个整体一起向系统提交或撤销操作请求，即这些操作要么同时成功，要么同时失败。

## 五.事务

### 一.概述

### 二.相关操作

- (1) 查看事务提交方式 `SELECT @@autocommit;`
- (2) 设置事务提交方式 `SET @@autocommit = 0;`  
0 - 手动提交, 1 - 自动提交
- (3) 提交事务 `COMMIT;`
- (4) 回滚事务 `ROLLBACK;`
- (5) 开启事务  
方式一: `START TRANSACTION`  
方式二: `BEGIN`

### 三.四大特性 (ACID)

- (1) 原子性 (Atomicity) 事务是不可分割的最小操作单元，要么全部成功，要么全部失败。
- (2) 一致性 (Consistency) 事务完成时，必须使所有的数据都保持一致状态。
- (3) 隔离性 (Isolation) 数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环境下运行。
- (4) 持久性 (Durability) 事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

### 四.并发事务问题

- (1) 脏读 一个事务读到另外一个事务还没有提交的数据。
- (2) 不可重复读 一个事务先后读取同一条记录，但两次读取的数据不同，称之为不可重复读。
- (3) 幻读 一个事务按照条件查询数据时，没有对应的数据行，但是在插入数据时，又发现这行数据已经存在，好像出现了“幻影”。

### 五.事务隔离级别

- 1.作用 解决并发事务所引发的问题
- 2.分类
  - (1) Read uncommitted 存在 脏读、不可重复读、幻读 问题
  - (2) Read committed 存在 不可重复读、幻读 问题
  - (3) Repeatable Read(默认) 存在 幻读 问题
  - (4) Serializable 没问题，但是性能较差
- 3.相关操作
  - (1) 查看事务隔离级别 `SELECT @@TRANSACTION_ISOLATION;`
  - (2) 设置事务隔离级别 `SET [ SESSION | GLOBAL ] TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }`