# Mysql事务

## 事务的简介

#### 为什么需要事务

现在的很多软件都是多用户，多程序，多线程的，对同一个表可能同时有很多人在用，为保持数据的一致性，所以提出了事务的概念。

A 给B 要划钱，A 的账户-1000元， B 的账户就要+1000元，这两个update 语句必须作为一个整体来执行，不然A 扣钱了，B 没有加钱这种情况很难处理。

#### 什么存储引擎支持事务

1.查看数据库下面是否支持事务（InnoDB支持）？

show engines;

2.查看mysql当前默认的存储引擎？

show c'relike '%storage\_engine%';

3.查看某张表的存储引擎？

show create table 表名 ;

4.对于表的存储结构的修改？

建立InnoDB 表：Create table .... type=InnoDB； Alter table table\_name type=InnoDB;

#### 事务的特性

事务应该具有4个属性：原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

原子性（atomicity）。一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性（consistency）。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

隔离性（isolation）。一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

持久性（durability）。持久性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

原子性（atomicity）

一个事务必须被视为一个不可分割的最小单元，整个事务中的所有操作要么全部提交成功，要么全部失败，对于一个事务来说，不可能只执行其中的一部分操作

比如: 老婆大人给Deer老师发生活费

1.老婆大人工资卡扣除500元

2.Deer老师工资卡增加500

不能出现工资卡扣除500元，但Deer老师工资卡没有增加500元的情况

要么全部成功，要么全部失败

一致性（consistency）

一致性是指事务将数据库从一种一致性转换到另外一种一致性状态，在事务开始之前和事务结束之后数据库中数据的完整性没有被破坏

比如：

1.老婆大人工资卡扣除500元

2.Deer老师工资卡增加500

2.Deer老师工资卡增加1000

不能因为任何原因，导致Deer老师收到两次钱

持久性（durability）

一旦事务提交，则其所做的修改就会永久保存到数据库中。此时即使系统崩溃，已经提交的修改数据也不会丢失

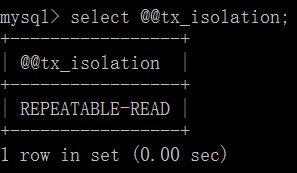
隔离性（isolation）

隔离性要求一个事务对数据库中数据的修改，在未提交完成前对于其他事务是不可见的

## 事务隔离级别

mysql默认的事务隔离级别为repeatable-read

show variables like '%tx\_isolation%';



### 未提交读（READ UNCOMMITED）脏读

set SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

**测试：**

启动两个session

一个session中

start TRANSACTION

update account set balance = balance -50 where id = 1

另外一个session中查询

select \* from account

回到第一个session中 回滚事务

ROLLBACK

在第二个session种

update account set balance = balance -50 where id = 1

查询结果还是 400

第二个session以为结果是350，但前面的400数据为脏读数据，导致最后的结果和意料中的结果并不一致。

### 已提交读 （READ COMMITED）不可重复读

**测试**

show variables like '%tx\_isolation%';

set SESSION TRANSACTION ISOLATION LEVEL read committed;

一个session中

start TRANSACTION

update account set balance = balance -50 where id = 1

另外一个session中查询 (数据并没改变)

select \* from account

回到第一个session中 回滚事务

commit

在第二个session种

select \* from account (数据已经改变)

### 可重复读（REPEATABLE READ）

**测试**

show variables like '%tx\_isolation%';

set SESSION TRANSACTION ISOLATION LEVEL repeatable read;

一个session中

start TRANSACTION

update account set balance = balance -50 where id = 1

另外一个session中查询 (数据并没改变)

select \* from account

回到第一个session中 回滚事务

commit

在第二个session种

select \* from account (数据并未改变)

### 可串行化（SERIALIZABLE）

account 表有3条记录，业务规定，最多允许4条记录。

**1.开启一个事务**

begin

select \* from account 发现3条记录

**2.开启另外一个事务**

begin

select \* from account 发现3条记录 也是3条记录

insert into account VALUES(4,'deer',500)

查询 4条记录

select \* from account

**3.回到第一个session**

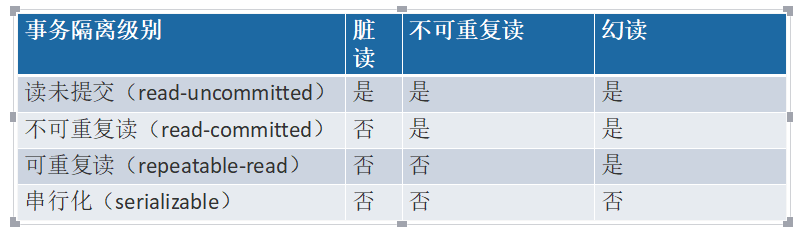
insert into account VALUES(5,'james',500)

select \* from account 4条记录

**4.session1 与 session2 都提交事务**

1. **set SESSION TRANSACTION ISOLATION LEVEL serializable; 重新上面的测试发现插入报错**

### 总结



**事务隔离级别为可重复读时，如果有索引（包括主键索引）的时候，以索引列为条件更新数据，会存在间隙锁间、行锁、页锁的问题，从而锁住一些行；如果没有索引，更新数据时会锁住整张表**

**事务隔离级别为串行化时，读写数据都会锁住整张表**

**隔离级别越高，越能保证数据的完整性和一致性，但是对并发性能的影响也越大，对于多数应用程序，可以优先考虑把数据库系统的隔离级别设为Read Committed，它能够避免脏读取，而且具有较好的并发性能。**

## 事务语法

### 开启事务

1、begin

2、START TRANSACTION（推荐）

3、begin work

### 事务回滚

rollback

### 事务提交

commit

### 还原点

savepoint

show variables like '%autocommit%'; 自动提交事务是开启的

set autocommit=0;

insert into testdemo values(5,5,5);

savepoint s1;

insert into testdemo values(6,6,6);

savepoint s2;

insert into testdemo values(7,7,7);

savepoint s3;

select \* from testdemo

rollback to savepoint s2

rollback