# 事务学习

为解决多事务并发的问题，数据库提供了事务隔离机制、锁机制、MVCC多版本并发控制隔离机制、日志机制。

## 事务四大特性ACID

1. A原子性：事务要不成功，要不失败。要通过undo log实现。
2. C一致性：多事务并发执行结果，必须以某一串行执行方式保持一致。
3. I隔离性：各并发事务执行互不影响。由锁机制、MVCC机制实现。
4. D持久性：提交以后就是永久性，由redo log实现。

## 并发处理导致的问题

1. 更新掉失/脏写：并发更新同一条数据导致更新掉失，例如秒杀系统的库存递减。
2. 脏读：事务A读取事务B已经更新但没有提交的数据。
3. 不可重复读：事务A在不同时刻读取不一样的数据。
4. 幻读：事务A读到事务B提交的新数据。

事务严谨性影响：脏读>不可重复读>幻读

## 事务隔离级别

脏读、不可重复读、幻读都可以通过事务隔离级别来解决。

### 隔离级别

1. Read-uncommitted：允许事务读取未提交的数据。此级别最不严格，也是最危险的。
2. Read-committed：只能读取已经提交的数据。这意味着在同一事务内，先后两次读取同一行数据结果可能不同。
3. repeatable-read：保证同一个事务中多次读取同一行数据时的结果始终相同。必须等待当前事务完成后才能提交新的修改。
4. Serializable：最严格的级别，保证同一个事务中多次读取同一行数据时的结果始终相同，并且防止任何并发更新。所有对事务期间读取的数据都将锁定，以确保不能对其进行修改。

### 并发问题处理

|  |  |  |  |
| --- | --- | --- | --- |
| 隔离级别 | 脏读 | 不可重复读 | 幻读 |
| 1读未提交(Read-uncommitted) | 可能 | 可能 | 可能 |
| 2读已提交(Read-committed) | 不可能 | 可能 | 可能 |
| 3可重复读(repeatable-read) | 不可能 | 不可能 | 可能 |
| 4串行(Serializable) | 不可能 | 不可能 | 不可能 |

1. 隔离级别:读未提交>读已提交>可重复读>串行
2. 隔离级别越高，成本越大。
3. mysql默认事务隔离级别是“可重复读”，oracle是“读已提交”
4. 可以通过spring设置，不设置按数据库默认；

### 命令

查看所有变量: SHOW VARIABLES;

查看事务级别：

show variables like 'transaction\_isolation';

设置隔离级别：

set transaction\_isolation='Read-uncommitted';

set transaction\_isolation='Read-committed';

set transaction\_isolation='repeatable-read';

set transaction\_isolation='Serializable';

### 读未提交测试

请看《05.3.1 Read-uncommitted-A.sql》、《05.3.2 Read-uncommitted-B.sql》

### 读已提交测试

请看《05.4.1 Read-committed-A.sql》、《05.4.2 Read-committed-B.sql》

### 可重复读测试

请看《05.5.1 repeatable-read-A.sql》、《05.5.2 repeatable-read-B.sql》

### 串行测试

请看《05.5.1 Serializable-A.sql》、《05.5.2 Serializable-B.sql》

## 读锁与写锁

Select \* from acccount lock in share mode; --读锁，这就是串行的原理

Select \* from acccount for update; --写锁

## 大事务操作危害

1. 并发的时候连接池会被撑爆
2. 锁太多的数据，导致大量阻塞和锁超时
3. 执行时间长，容易造成主从延迟
4. 回滚耗时长
5. Undo log日志爆满
6. 容易死锁

## 事务优化

1. 查询语句不放在事务；
2. 事务避免远程调用，或设置超时时间避免事务等待过久
3. 避免大事务，可拆分多事务执行
4. 加锁动作放在事务最后执行
5. 能异步尽量异步
6. 应用则保证最终一致性，而不是事务

## 强制结束事务

查询超过1S的事务

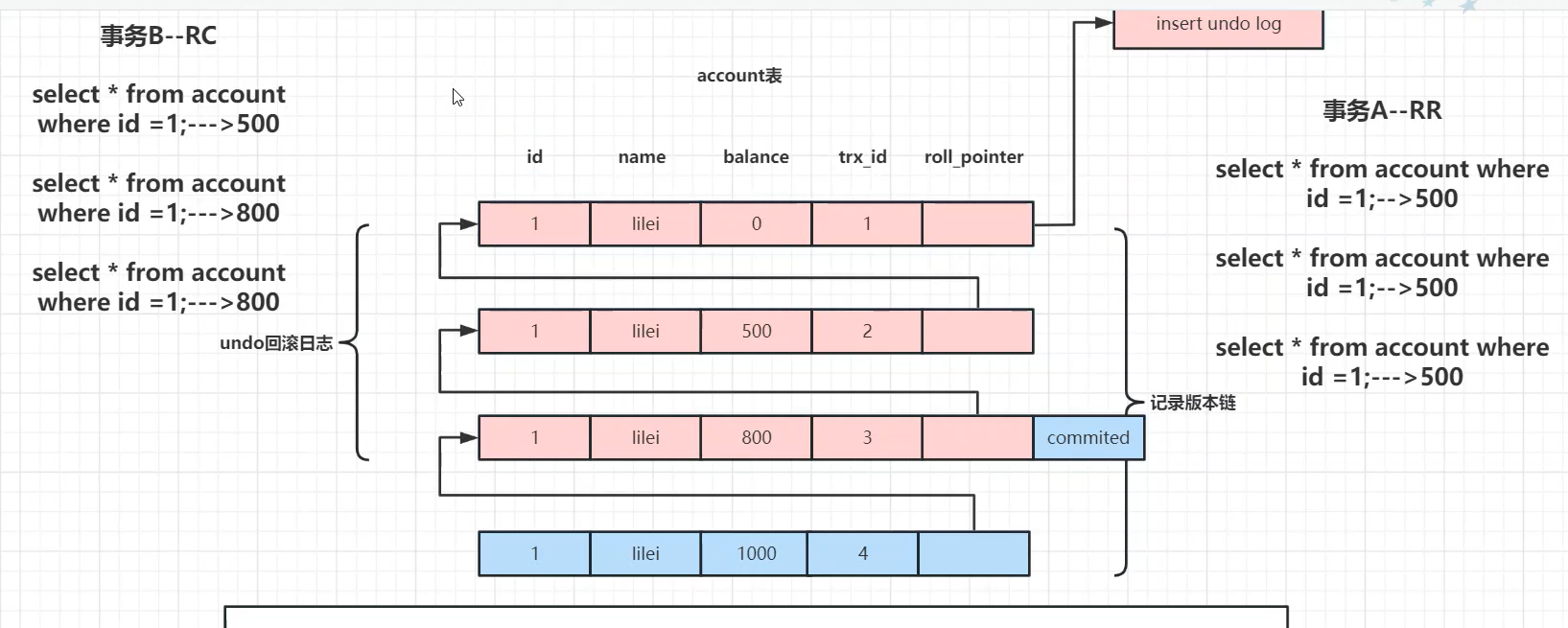
SELECT \* FROM information\_schema.innodb\_trx WHERE TIME\_TO\_SEC( timediff( now(), trx\_started)) >1;

杀死事务

Kill id

## 浅谈mvcc

Mvcc，多版本并发控制机制。



Mysql操作包含很多日志表，其中包含undo回滚日志，每一次更新操作都会插入到该表中。该表结构除了原始表字段，还有事务id(trx\_id)、上一条记录指针(rollback\_pointer)。

事务的提交、回滚就是根据这两个字段记录。