并发问题

1 脏读（等于无法隔离，mysql应该禁止）

事务a 事务b

Insert 1:100

Update 1:200

Select（1:200）

Rollback

Selcet(1:100)

事务a读取到了事务b修改的数据

2 不可重复读（没完全隔离,但是可以用）

事务a 事务b

Insert 1:100

Update 1:200

Select（1:100）

commit

Select(2:100)

事务a虽然不会读取事务b的没comimi的数据，但是事务bcommit后，事务a会读取到。

3 幻读(id,content)

事务a 事务b

Select (1,a,2,b,3,c )3条数据

Insert 4,d  （插入第4条数据）

再次查询也是查询到

（1a,2b,3c）3条数据

Insert 4,e的时候，此时会报主键(4,e和4,d)冲突

此时无论再怎么样查询，都是只有

（1a,2b,3c）3条数据

4种隔离级别

Read uncommited 读未提交 但会存在脏读

Read commited 读已提交 事物过程中会读取到其他事务已经提交的数据，但会存在不可重复读现象

Repeatable read 可重复读 每次读取相同结果集，不管其他事物是否已提交。但会存在幻读现象

Serializable 串行化 事物排队 隔离级别最高，性能最差

事务实现原理

Mvcc

1 多版本并发控制

* 1. 当前读

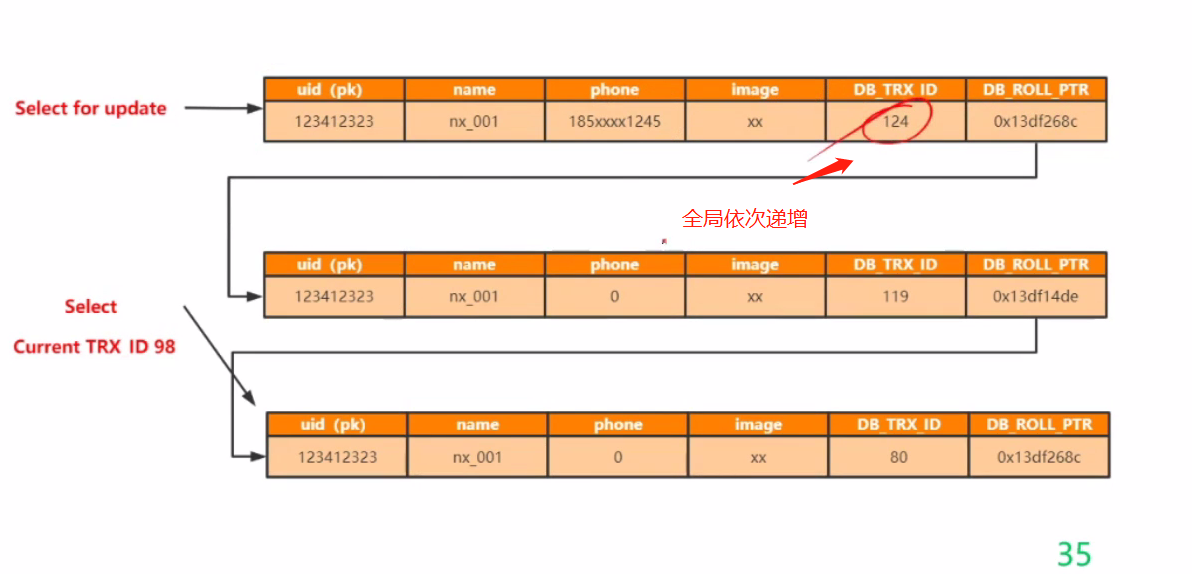
Select for update （读最新update的数据,一般不用）

* 1. 快照读

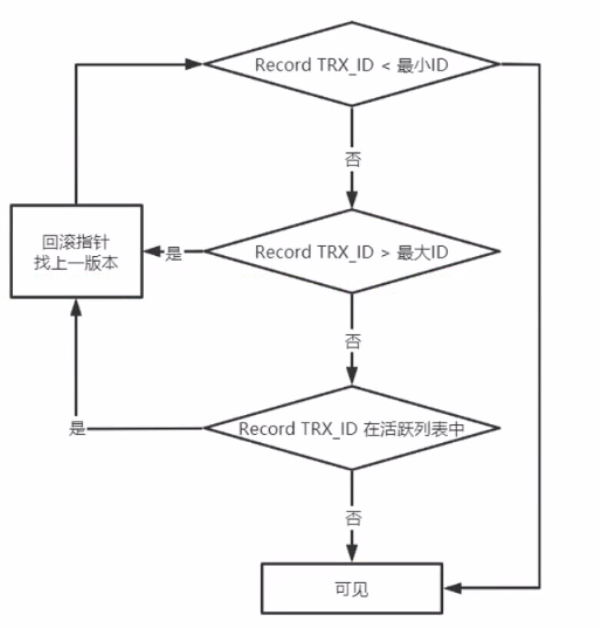
简单的select都是快照读

读的是某一个历史版本

在一个事务内，生成一个快照，每次读都是读这个历史版本。就可以做到了可重复读



Mvcc在RR（可重复读下）的判断流程



* + 1. 快照读详细情况

当select的时候，系统就自动创建了一条trx\_id（假设是n）

1.2.1.1如果还有没有提交的事务存在，那读的是老的数据(防止事务回滚)。哪怕那些事务提交之后，他依旧读的是当前读的数据。

1.2.1.2如果在trx\_id之后的事务数据，无论事务是否已经提交，他读不到

* + 1. 系统内可见的活跃的trx\_id列表（未提交事务的id）

假设有列表 [8,10,16,33,191]

最小id是8 最大id是220(全局要递增的id，假设之前191-220之间的数据已经提交)

* + 1. 寻找快照读（版本）的策略

如上图，假设去到数据库，开始寻找事务x = record trx id

record trx id的值比最小值还小，那这条记录就是可见的（因为比8还小说明事务已经提交）.

如不是，则判断是否比当前最大的id还要大，如果是的话，那么当前读的这条数据就要回滚指针到上一条记录，再重新做判断.依次循环，一直到record trx id不在事务列表中，数据才是当前读的数据。

其实也就是一直寻求比最小值还小的事务id作为快照读

Undo log

回滚日志,保证事务的原子性,实现数据多版本也就是mvcc存放在undo log！！

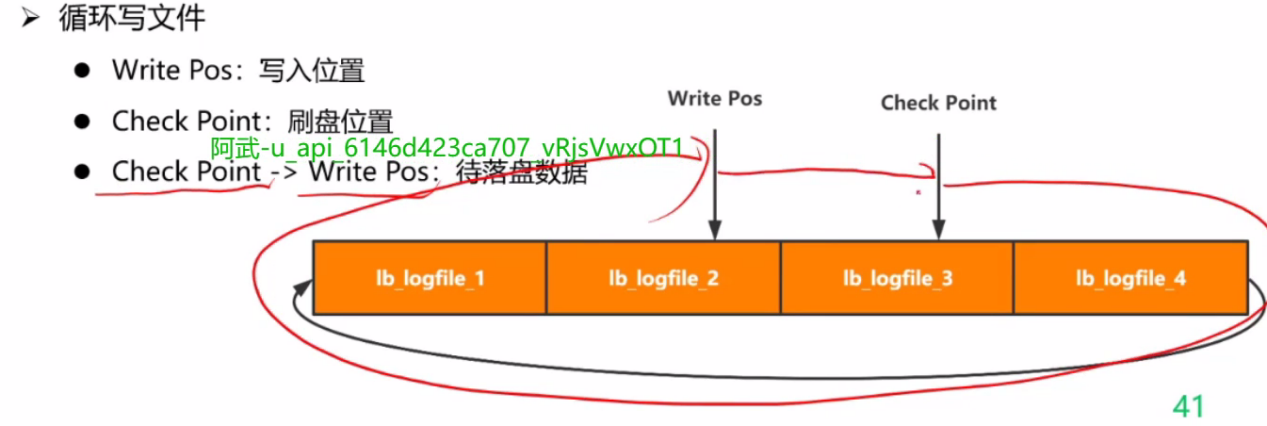
Delete undo log 用于回滚，提交即清理

Update undo log 用于回滚，同时实现快照读，不能随便删除

Redo log

保证事务的持久性

用于comiit时候，失败的时候，再次redo一下.如图所示：



整体流程图

