**MVCC**

**什么是MVCC**

mvcc 全程multi-version-concurrency-control,多版本并发控制，是MySQL事务用来解决事务读写之间冲突的方案，该机制实现了不同的事务可以对同一份数据同时进行读写，mvcc工作在RC和RR隔离级别下，并且RC和RR级别下的MVCC略微有点区别。

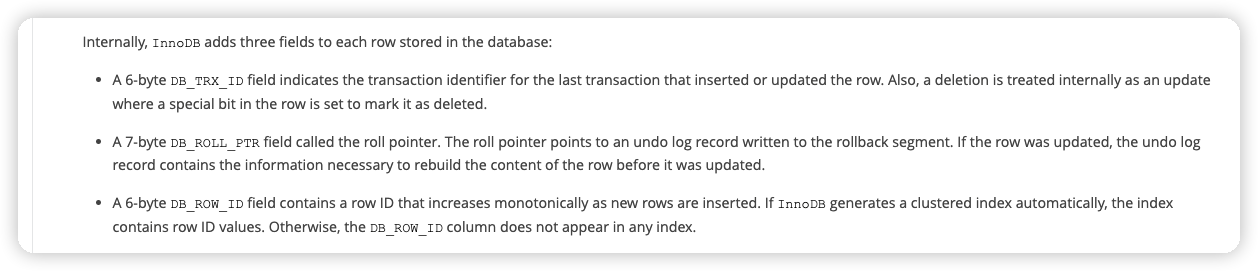
**MVCC实现逻辑**

mvcc 的实现涉及到MySQL 对数据库表加的一些隐藏字段：

DB\_TRX\_ID(6 bytes): 事务id，单调递增

DB\_ROW\_PTR(7 bytes)：回滚指针,指向上次undolog 版本链上上一次的记录

DB\_ROW\_ID(6 bytes)：当一条新的记录插入时，MySQL自动为该记录生成的一个递增的id



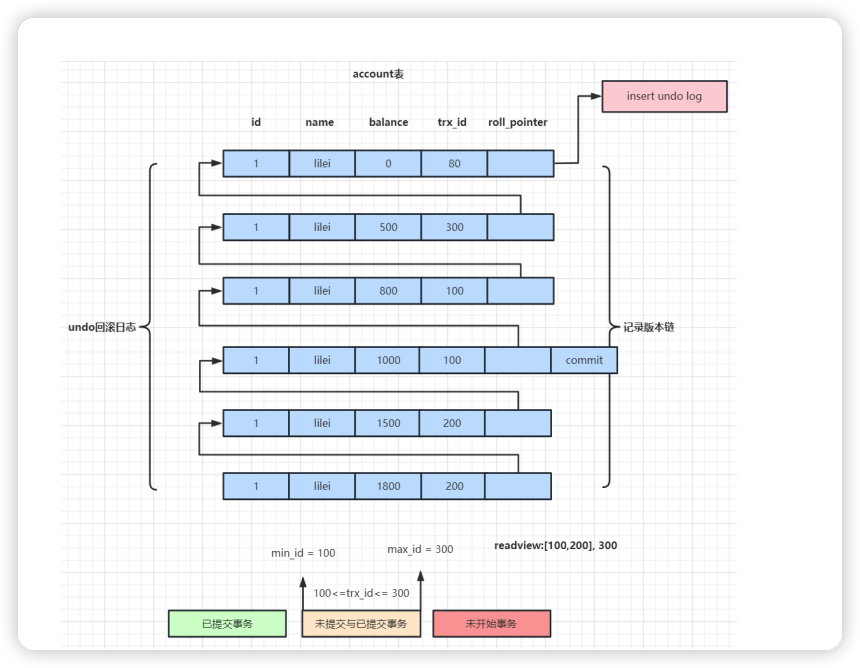
MySQL 的MVCC 想要实现，还需要一个很重要的东西，就是一致性视图。

一致性视图：RR 级别在事务开启后，第一次查询时生成，直到该事务提交前一致性视图都不会在改变；RC级别，**在每一次执行查询时都会重新生成一致性视图**。通过一致性视图结合可见性算法就实现了mvcc机制。

**MVCC可见性算法**

mvcc的可见性算法you以下几部分组成:

undolog 回滚链：当你在MySQL中执行一个update/insert/delete 时，会对修改前的版本进行记录，同时用最新的记录的DB\_ROW\_PTR字段指向修改前的记录；



活跃事务id数组：ids[]；

活跃事务id中最小的事务id：minId；

当前MySQL中已创建的事务id中最大的事务id：maxId；

这几个元素将整个一致性视图中的事务拆分成了3部分：

小于minId的事务：代表这些事务在生成一致性视图前就已经提交，是可见的

大于maxId的事务：代表这些事务是在一致性视图创建后才创建的，不可见

大于等于minId,小于等于maxId：处于这之间的事务id得分两种情况:

a. 事务id在活跃事务id中，代表该事务还未提交，不可见

b. 事务id不在活跃事务id中，代表该事务已经提交，可见

**一致性视图的使用**：查询数据时，从undo log 回滚链从最新的数据开始，依次往回对比，直到找到符合可见性算法的数据为止。

**注意点**

当执行begin/start transaction时并不会立即生成事务id,事务id的生成实际在第一次执行修改操作或加排他锁(select ... for update)时; 当开启事务后第一次执行 select 操作时，也会生成事务id,但此时的事务id时临时事务id,在后面继续执行修改操作或加锁操作后，该事务id会被更新。

**已提交事务的undolog 什么时候清除呢？**

已提交事务的undolog 是由MySQL系统级的purge线程完成的，由MySQL自己决定在系统负载比较低的时候进行。