InnoDB MVCC 多版本并发机制

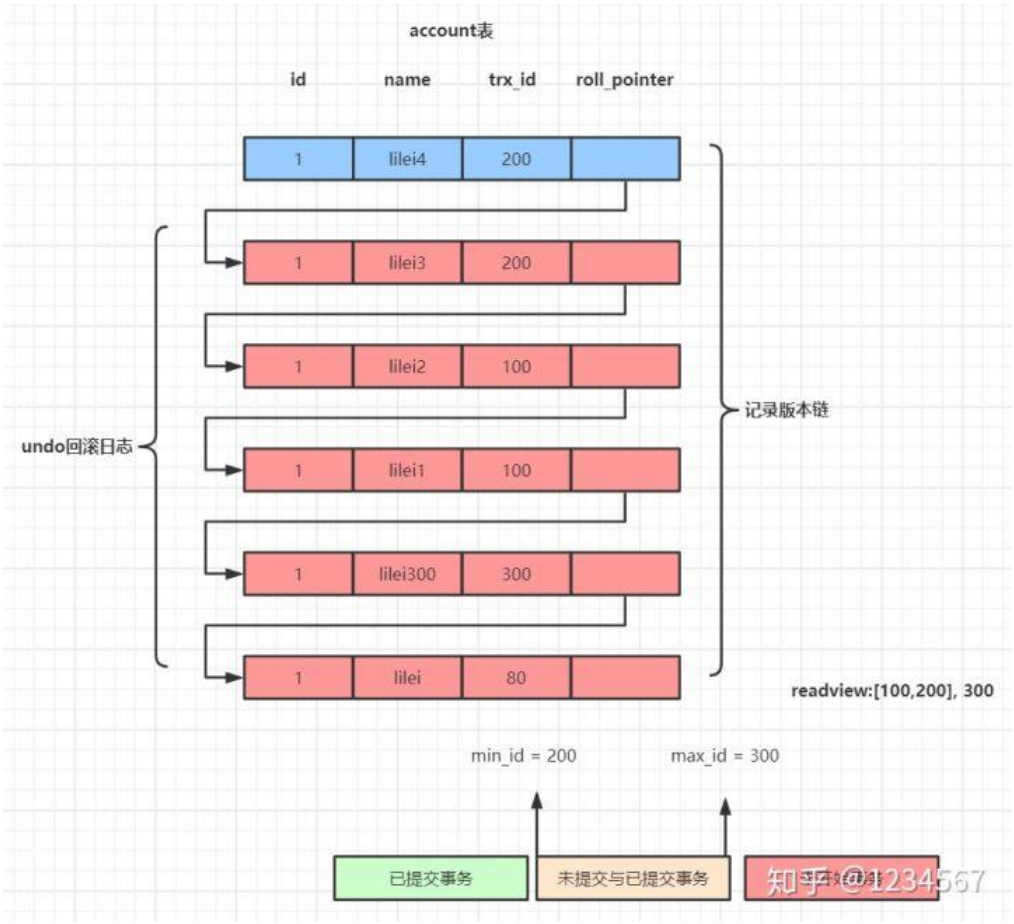
一、MySQL 4 种隔离级别

隔离级别	读数据一致性	脏读	不可重复读	幻读
未提交读(READ UNCOMMITTED)	最低级别，不读物理上顺坏的数据	是	是	是
已提交读(READ COMMITTED)	语句级	否	是	是
可重复读(REPEATABLE RED)	事务级	否	否	是
可序列化(SERIALIZABLE)	最高级别，事务级	否	否	否

READ_UNCOMMITTED 以及 SERIALIZABLE 级别不会使用到 MVCC 机制。
在读已提交和可重复读隔离级别下都实现了 MVCC 机制。
MVCC 在 MYSQL InnoDB 中的实现主要是为了提高数据库并发性能，用更好的方式去处理读-写冲突，做到即使有读写冲突时，也能做到不加锁，非阻塞并发读

- 脏读：指当一个事务正在访问数据，并且对数据进行了修改，而这种数据还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个数据。因为这个数据还没有提交那么另外一个事务读取到的这个数据我们称之为脏数据。
- 不可重复读：指在一个事务内，多次读同一数据。在这个事务还没有执行结束，另外一个事务也访问同一数据，那么在第一个事务中的两次读取数据之间，由于第二个事务的修改，第一个事务两次读到的数据可能是不一样的，这样就发生了在一个事物内两次连续读到的数据是不一样的，这种情况被称为是不可重复读。
- 幻读：一个事务先后读取一个范围的记录，但两次读取的纪录数不同，称之为幻读（两次执行同一条 select 语句会出现不同的结果，第二次读会增加一数据行）

二、MVCC



1、Undo log 版本链。每次有其他事务修改该条数据时，会把历史都保存下来，连成一条链表。

2、加入的新字段

a) roll_ptr 是链表指针，指向上一个版本。

b) TRX_ID，更新该行的事务 id

c) row_id，聚簇索引的主键，当没有设置主键时且没有设置 unique 索引时，该列充当主键（InnoDB 是必须有主键的）。

d) DELETE BIT 逻辑删除

三、快照读

1、快照（snapshot）就是数据库某时刻的一个状态。

a) 对于 READ_COMMITTED：对于每个操作都需要新建一个 snapshot，因为有些事务可能在该事务运行过程中提交。

b) 对于 REPEATABLE_READ:对于每个新事务只需要在该事务开始执行时新建一个 snapshot 即可，以后读取的数据将不会变动

2、快照里有 4 个重要的变量

a) creator_trx_id 当前事务 id

b) up_limit_id 当前在线事务 id 的最小值（最先开始的）

c) low_limit_id 当前在线事务 id 的最大值 + 1（最后开始的），+1 后区间是左闭右开。

d) trx_ids 当前正在运行的事务列表。是当前快照中还未提交的事务。

3、在上面的版本链中找到事务应当读取的数据

TRX_ID 是存表里的，更新该行的事务 id。

a) 若 TRX_ID < up_limit_id，说明更新该行的事务 id 比当前在线事务 id 的最小值小，说明该事务已提交，该记录可以读到。
b) 如果 TRX_ID >= low_limit_id，则表明该事务在创建 Read View 之后才开始，该条记录对于当前事务而言不可见。
c) 如果 up_limit_id <= TRX_ID < low_limit_id，此时需要在活跃事务链表中查找是否存在 ID 为 TRX_ID 的值的事务，即 TRX_ID 是否在 trx_ids 中
   i. 在，说明该事务未提交，则该条记录不可见。
   ii. 不在，说明事务已提交，该条记录可见。
d) 若该记录不可见，则沿着链表查找下一条记录，直到找到可见的记录。

4、更新快照

a) 对于 Read Committed 级别的
   每次执行 select 都会创建新的 read_view.
b) 对于 Repeatable Read 级别的
   第一次 select 时更新这个 read_view，以后不会再更新，后续所有的 select 都是复用这个 read_view。

四、当前读和快照读

a) 当前读是读取当前数据库的最新版本，当前读通过 next-key 锁（行锁+gap 锁）来解决幻读问题的。
b) 快照读读取的是快照，不需要加锁也能解决幻读问题。因为其他事务插入新数据时，TRX_ID 大于当前事务 ID 的数据记录，不会被当前事务读到，解决了幻读问题。

有资料说 MVCC 不能完全解决幻读问题，但我没有看懂：

如果你在一个事务中先查询了一个数据，然后插入或者更新相关的数据，这个时候来了一个事务 B 同时更新或者删除你要查询的记录，就会出现幻读问题了。

五、增删改查：

①增加：INSERT

- 设置新记录的 TRX_ID 为当前事务 ID，其他的采用默认的。

②删除：DELETE

- 修改 TRX_ID 的值为当前的执行删除操作的事务的 ID，然后设置 DELETE BIT 为 True，表示被删除

③修改：UPDATE <==> INSERT + DELETE

- 用 X 锁锁定该行（因为是写操作）；

- 记录 redo log：将更新之后的数据记录到 redo log 中，以便日后使用；
- 记录 undo log：将更新之后的数据记录到 undo log 中，设置当前数据行的 DATA_TRX_ID 为当前事务 ID，回滚指针 DATA_ROLL_PTR 指向 undo log 中的当前数据行更新之前的数据行，同时设置更新之前的数据行的 DATA_TRX_ID 为当前事务 ID，并且设置 DELETE BIT 为 True，表示被删除。

④查找：SELECT

- 如果当前数据行的 DELETE BIT 为 False，只查找版本早于当前事务版本的数据行(也就是数据行的 DATA_TRX_ID 必须小于等于当前事务的 ID)，这确保当前事务读取的行都是事务之前已经存在的，或者是由当前事务创建或修改的行；
- 如果当前数据行的 DELETE BIT 为 True，表示被删除，那么只能返回 DATA_TRX_ID 的值大于当前事务的行。获取在当前事务开始之前，还没有被删除的行。