

## 53 理解MVCC机制的前奏：undo log版本链是个什么东西？

### 理解MVCC机制的前奏：undo log版本链是个什么东西？

今天我们正式开始切入讲解MySQL中多个事务并发执行时的隔离到底是怎么做的，因为我们知道默认是骚气的RR隔离级别，也就是说脏写、脏读、不可重复读、幻读，都不会发生，每个事务执行的时候，跟别的事务压根儿就没关系，甭管你别的事务怎么更新和插入，我查到的值都是不变的，是一致的！

但是这到底是怎么做到的呢？

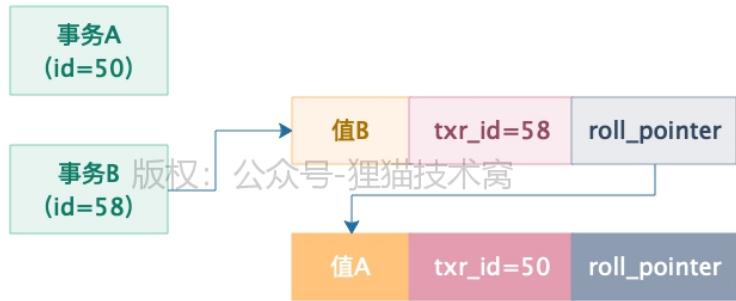
这就是由经典的**MVCC多版本并发控制机制**做到的，但是讲解这个MVCC机制之前，我们还得先讲讲undo log版本链的故事，这是一个前奏，了解了这个机制，大家才能更好的理解MVCC机制。

简单来说呢，我们每条数据其实都有两个隐藏字段，一个是trx\_id，一个是roll\_pointer，这个trx\_id就是最近一次更新这条数据的事务id，roll\_pointer就是指向你了你更新这个事务之前生成的undo log，关于undo log之前都讲过了，这里不用多说了。

我们给大家举个例子，现在假设有一个事务A (id=50)，插入了一条数据，那么此时这条数据的隐藏字段以及指向的undo log如下图所示，插入的这条数据的值是值A，因为事务A的id是50，所以这条数据的trx\_id就是50，roll\_pointer指向一个空的undo log，因为之前这条数据是没有的。



接着假设有一个事务B跑来修改了一下这条数据，把值改成了值B，事务B的id是58，那么此时更新之前会生成一个undo log记录之前的值，然后会让roll\_pointer指向这个实际的undo log回滚日志，如下图所示。

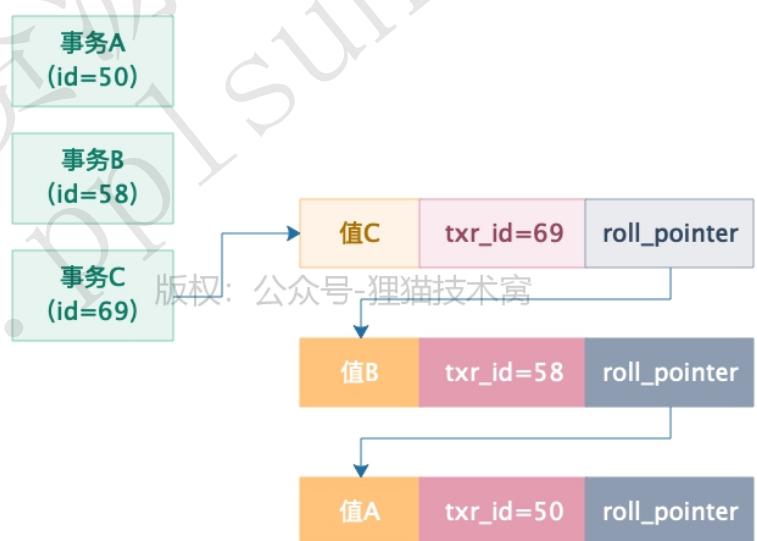


大家看上图是不是觉得很有意思？事务B修改了值为值B，此时表里的那行数据的值就是值B了，那行数据的txr\_id就是事务B的id，也就是58，roll\_pointer指向了undo log，这个undo log就记录你更新之前的那条数据的值。

所以大家看到roll\_pointer指向的那个undo log，里面的值是值A，txr\_id是50，因为undo log里记录的这个值是事务A插入的，所以这个undo log的txr\_id就是50，我还特意把表里的那行数据和undo log的颜色弄成不一样的，以示区分。

接着假设事务C又来修改了一下这个值为值C，他的事务id是69，此时会把数据行里的txr\_id改成69，然后生成一条undo log，记录之前事务B修改的那个值

此时如下图所示，看起来如下。



我们在上图可以清晰看到，数据行里的值变成了值C，txr\_id是事务C的id，也就是69，然后roll\_pointer指向了本次修改之前生成的undo log，也就是记录了事务B修改的那个值，包括事务B的id，同时事务B修改的那个undo log还串联了最早事务A插入的那个undo log，如图所示，过程很清晰明了。

所以这就是今天要给大家讲的一点，大家先不管多个事务并发执行是如何执行的，起码先搞清楚一点，就是多个事务串行执行的时候，每个人修改了一行数据，都会更新隐藏字段txr\_id和roll\_pointer，同时之前多个数据快照对应的undo log，会通过roll\_pinter指针串联起来，形成一个重要的版本链！

今天要让大家明白的，就是这个多个事务串行更新一行数据的时候，txr\_id和roll\_pinter两个隐藏字段的概念，包括undo log串联起来的多版本链条的概念！

**End**