Atitit update 执行流程与事务

UPDATE语句修改该行数据时，

会首先使用排他锁锁定改行，

将该行当前的值复制到undo log中，

然后再真正地修改当前行的值，

最后填写事务ID，

使用回滚指针指向undo log中修改前的行。如下图所示。

2、记录行修改的具体流程

① 首先当前事务对记录行加排他锁

② 然后把改行数据拷贝到undo log中，作为旧版本

③ 拷贝完毕后，修改该行的数据

④ 事务提交，提交前用 CAS 机制判断记录行当前最新修改的事务id 是否发生了变化，如果没变，则修改记录行最新的修改事务id ，也就是DB\_TRX\_ID为当前事务id，并提交，如果变了，说明存在其他事务修改了这个记录行，那么就应该回滚这个事务。也就是当前事务没有生效。

MVCC逻辑流程-插入

在MySQL中建表时，每个表都会有三列隐藏记录，其中和MVCC有关系的有两列

数据行的版本号 （DB\_TRX\_ID）

删除版本号 (DB\_ROLL\_PT)

id

test\_id

DB\_TRX\_ID

DB\_ROLL\_PT

在插入数据的时候，假设系统的全局事务ID从1开始，以下SQL语句执行分析参考注释信息：

begin;-- 获取到全局事务ID

insert into `test\_zq` (`id`, `test\_id`) values('5','68');

insert into `test\_zq` (`id`, `test\_id`) values('6','78');

commit;-- 提交事务

复制代码当执行完以上SQL语句之后，表格中的内容会变成：

id

test\_id

DB\_TRX\_ID

DB\_ROLL\_PT

5

68

1

NULL

6

78

1

NULL

可以看到，插入的过程中会把全局事务ID记录到列 DB\_TRX\_ID 中去

MVCC逻辑流程-删除

对上述表格做删除逻辑，执行以下SQL语句（假设获取到的事务逻辑ID为 3）

begin；--获得全局事务ID = 3

delete test\_zq where id = 6;

commit;

复制代码执行完上述SQL之后数据并没有被真正删除，而是对删除版本号做改变，如下所示：

id

test\_id

DB\_TRX\_ID

DB\_ROLL\_PT

5

68

1

NULL

6

78

1

3

MVCC逻辑流程-修改

修改逻辑和删除逻辑有点相似，修改数据的时候 会先复制一条当前记录行数据，同事标记这条数据的数据行版本号为当前是事务版本号，最后把原来的数据行的删除版本号标记为当前是事务。

执行以下SQL语句：

begin;-- 获取全局系统事务ID 假设为 10

update test\_zq set test\_id = 22 where id = 5;

commit;

复制代码执行后表格实际数据应该是：

id

test\_id

DB\_TRX\_ID

DB\_ROLL\_PT

5

68

1

10

6

78

1

3

5

22

10

NULL

MVCC逻辑流程-查询

此时，数据查询规则如下：

查找数据行版本号早于当前事务版本号的数据行记录

也就是说，数据行的版本号要小于或等于当前是事务的系统版本号，这样也就确保了读取到的数据是当前事务开始前已经存在的数据，或者是自身事务改变过的数据

查找删除版本号要么为NULL，要么大于当前事务版本号的记录

这样确保查询出来的数据行记录在事务开启之前没有被删除

根据上述规则，我们继续以上张表格为例，对此做查询操作

作者：程序猿杂货铺

链接：https://juejin.cn/post/6844903778026536968

来源：掘金

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。