Atitit 事务与undo机制

**If noUndo,,alredy ok**

**If noUndo,,alredy ok**

**If noUndo,,alredy ok**

6.6.3 Undo Log不是Log

了解Undo Log的功能后，进一步来看Undo Log的结构。其实Undo Log这个词有很大的迷惑性，它其实不是Log，而是数据。为什么这么说？

（1）Undo Log并不像Redo Log一样按照LSN的编号，从小到大依次执行append操作。Undo Log其实没有顺序，多个事务是并行地向Undo Log中随机写入的。

（2）一个事务一旦Commit之后，数据就“固化”了，固化之后不可能再回滚。这意味着Undo Log只在事务Commit过程中有用，一旦事务Commit了，就可以删掉UndoLog。具体来说：

对于insert记录，没有历史版本数据，因此insert的Undo Log只记录了该记录的主键ID，当事务提交之后，该Undo Log就可以删除了；

下面来看这个“备份数据”是怎么操作的。如图6-18所示，Page中的每条记录，除了自身的主键ID和数据外，还有两个隐藏字段：一个是修改该记录的事务ID，一个是rollback\_ptr，用来串联所有的历史版本。假设该记录被tx\_id为68、80、90、100的四个事务修改了四次，该数据就有四个版本，通过rollback\_ptr从新到旧串联起来。

6.6.4 Undo Log与Redo Log的关联

Undo Log本身也要写入磁盘，但一个事务修改多条记录，产生多条Undo Log，不可能同步写入磁盘，所以遇到了开篇讲Write-Ahead时的问题。如何解决Undo Log需要多次写入磁盘的效率问题呢？

Redo Log记录的是对数据的修改，凡是对数据的修改，都必须记入Redo Log。可以把Undo Log也当作数据！在内存中记录Undo Log，异步地刷盘，宕机重启，用Redo Log恢复Undo Log。

拿一个事务来举例：

start transaction update表1某行记录 delete表1某行记录 insert表2某行记录 commit

把Undo Log和Redo Log加进去，此事务类似下面伪代码所示：

start transaction 写Undo Log1: 备份该行数据（update） update表1某行记录 写Redo Log1 Undo Log2:备份该行数据（insert） delete 表1某行记录 写Redo Log2 Undo Log3:该行的主键ID（delete） insert表2某行记录 写Redo Log3 commit

在这里，所有Undo Log和Redo Log的写入都可以只在内存中进行，只要保证Commit之后Redo Log落盘即可，Undo Log可以一直保留在内存里，之后异步刷盘。

文章太长，未完待续，接下来的1篇，会继续分析Undo Log。