**MySQL 日志：undo log、redo log、binlog 有什么用？**

**执行一条 update 语句**会和select语句走类似的流程。

更新语句的流程会涉及到 undo log（回滚日志）、redo log（重做日志） 、binlog （归档日志）这三种日志：

* **undo log（回滚日志）**：Innodb 存储引擎层生成的日志，实现了事务中的**原子性**，主要**用于事务回滚和 MVCC**。
* **redo log（重做日志）**：Innodb 存储引擎层生成的日志，实现了事务中的**持久性**，主要**用于掉电等故障恢复**；
* **binlog （归档日志）**：Server 层生成的日志，主要**用于数据备份和主从复制**；

## 为什么需要 undo log？

undo log 是一种用于撤销回退的日志。在事务没提交之前，MySQL 会先记录更新前的数据到 undo log 日志文件里面，当事务回滚时，可以利用 undo log 来进行回滚。

每当 InnoDB 引擎对一条记录进行操作（修改、删除、新增）时，要把回滚时需要的信息都记录到 undo log 里，比如：

* 在**插入**一条记录时，要把这条**记录的主键值**记下来，这样之后回滚时只需要把这个主键值对应的记录**删除**就好了；
* 在**删除**一条记录时，要把这条**记录中的内容都**记下来，这样之后回滚时再把由这些内容组成的记录**插入**到表中就好了；
* 在**更新**一条记录时，要把**被更新的列的旧值**记下来，这样之后回滚时再把这些列**更新为旧值**就好了。

在发生回滚时，就读取 undo log 里的数据，然后做原先相反操作（蓝色字）

一条记录的每一次更新操作产生的 undo log 格式都有一个 **roll\_pointer 指针**和一个 **trx\_id 事务id**：

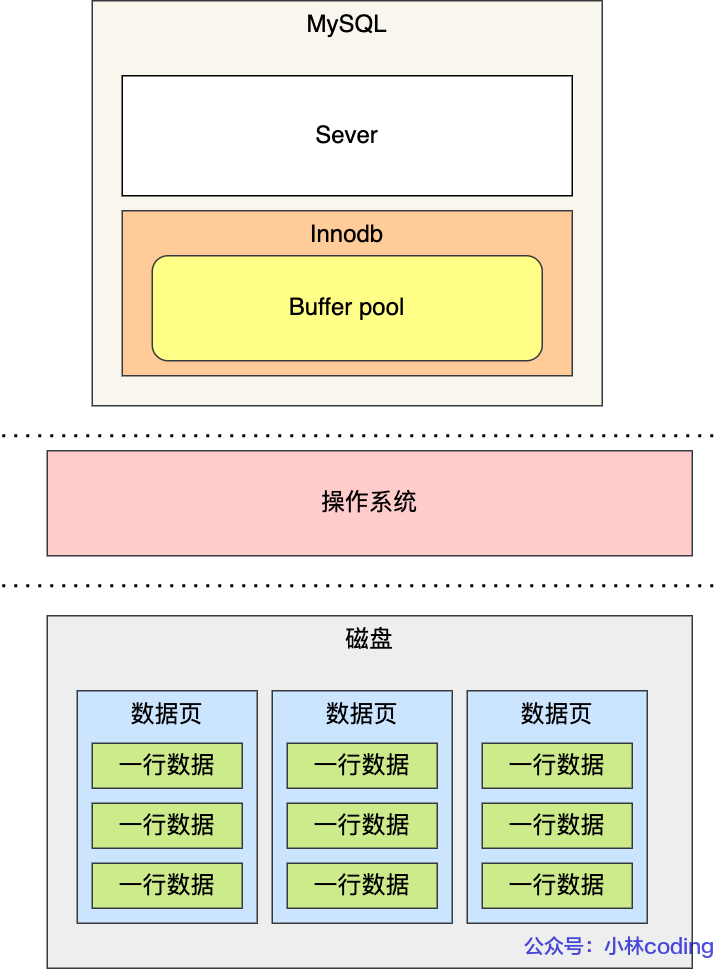
* 通过 trx\_id 可以知道该记录是被哪个事务修改的；
* 通过 roll\_pointer 指针可以将这些 undo log 串成一个链表，这个链表就被称为版本链；

undo log 两大作用：

* **实现事务回滚，保障事务的原子性**。事务处理过程中，如果出现了错误或者用户执 行了 ROLLBACK 语句，MySQL 可以利用 undo log 中的历史数据将数据恢复到事务开始之前的状态。
* **实现 MVCC（多版本并发控制）关键因素之一**。MVCC 是通过 ReadView + undo log 实现的。undo log 为每条记录保存多份历史数据，MySQL 在执行快照读（普通 select 语句）的时候，会根据事务的 Read View 里的信息，顺着 undo log 的版本链找到满足其可见性的记录。

## 为什么需要 Buffer Pool？

Innodb 存储引擎设计了一个**缓冲池（Buffer Pool）**，来提高数据库的读写性能。

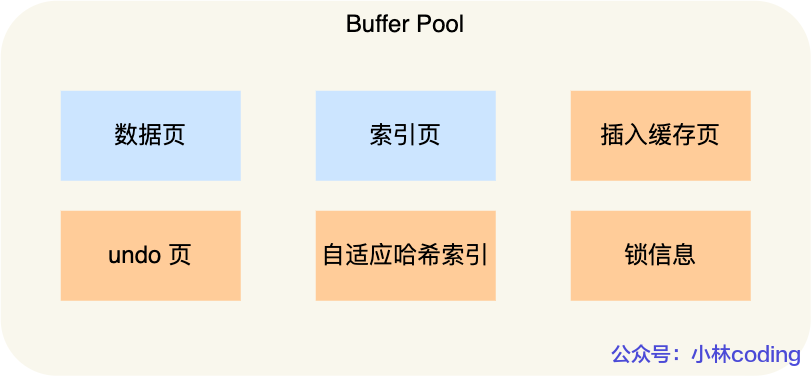


* 当**读取数据**时，如果数据存在于 Buffer Pool 中，客户端就会直接读取 Buffer Pool 中的数据，否则再去磁盘中读取。
* 当**修改数据**时，如果数据存在于 Buffer Pool 中，那直接修改 Buffer Pool 中数据所在的页，然后将其设置为**脏页**，为了减少磁盘I/O，**不会立即将脏页写入磁盘，而是由后台线程选择一个合适的时机将脏页写入到磁盘**。

### Buffer Pool 缓存什么？

Buffer Pool 同样需要按「页」来划分。MySQL 启动的时候，**InnoDB 会为 Buffer Pool 申请一片连续的内存空间，然后按照默认的16KB的大小划分出一个个的页， Buffer Pool 中的页就叫做缓存页**。查询一条记录时，InnoDB 是会**把整个页的数据加载到 Buffer Pool 中**，将页加载到 Buffer Pool 后，再通过页里的「页目录」去定位到某条具体的记录。

Buffer Pool的内容：



## 为什么需要 redo log ？

**WAL 技术： MySQL 的写操作并不是立刻写到磁盘上，而是先写日志，然后再在合适的时间再写到磁盘上**。

为了防止**断电**导致数据丢失的问题，当有一条记录需要更新的时候，InnoDB 引擎就会先更新内存（同时标记为脏页），然后**将本次对这个页的修改以 redo log 的形式记录下来**。后续InnoDB 引擎会在适当的时候，由后台线程将缓存在 Buffer Pool 的脏页刷新到磁盘里，这就是 **WAL （Write-Ahead Logging）技术**。