<https://mp.weixin.qq.com/s/yYToh_lWaETXmNqhcfRFlQ>

Innodb自增原理

虽然我们习惯于给主键ID指定AUTO\_INCREMENT属性，但是AUTO\_INCREMENT也是可以指定到非主键字段的，唯一的约束就是这个字段上面得加索引，有了索引，就可以通过类似SELECT MAX(\*ai\_col\*)的语句快速读到这列数据的最大值。

本文要探讨的话题是MySql的InnoDB引擎处理自增数据列的原理

**MySql 5.1之前的实现**

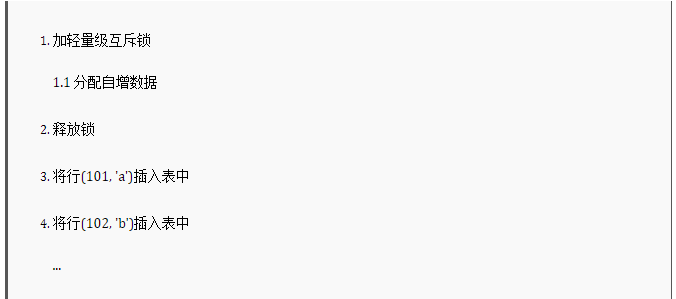
在这个版本之前，用AUTO\_INCREMENT修饰的数据列确实是严格连续自增的。MySql的实现是会针对每个插入语句加一个全表维度的锁，这个锁可以保证每次只有一条插入语句在执行，每插入一行数据，就会生成一个自增数据。



MySql5.1之前的这种实现方式可以保证AUTO\_INCREMENT严格自增，但是并发程度也最差，因为AUTO\_INCREMENT锁是全表加锁直到这条语句结束

前文中的insert语句是比较简单的，所谓简单的insert语句指的是插入的的数据行数是可以提前确定的，与之相对的是Bulk insert比如INSERT ... SELECT这类语句，这类插入语句的插入行数不能提前确定。

在这个版本以及之后，对于简单语句的插入，不再加全表的AUTO-INC锁，只会在产生自增列数据的时候加一个轻量级的互斥锁，等自增数据分配好，锁就释放了，因此像上面的例子，在MySql5.1之后的执行流程如下



可以看到，对于简单的插入语句，并发情况下的临界区变小了，且不再持有全表的锁，提升了并发性能。当然，如果在尝试加锁的过程中遇到有其他事务持有全表的AUTO-INC锁，还是要等待全表的AUTO-INC锁释放再执行本次插入操作

对于Bulk insert的插入语句，仍然避免不了全局的AUTO-INC锁，这类语句，他们的执行流程仍然保持和5.1之前版本一致。

## MySql 8.0版本之后的优化

虽然MySql5.1版本对简单的插入语句做了优化，避免了全表加锁，但对于INSERT ... SELECT这样的复杂插入语句，仍然避免不了全表的AUTO-INC锁，主要是基于执行语句的主从复制要能在从库完全回放复制主库，所有的语句执行结果就不能和执行顺序有关。

在MySql 8.0以及之后默认的主从复制策略变成了基于数据行实现，在这样的背景下INSERT ... SELECT这样的复杂插入语句也不需要全表加锁来生成自增列数据了，所有的插入语句只有在生成自增列数据的时候要求持有一个轻量级的互斥锁，等到自增数据生成好之后释放锁。在这种实现下，所有插入语句的自增列都不能保证连续自增，但是并发性能确实最好的。

**总结**

需要说明的是，如果插入语句所处的事务回滚了，生成的自增列数据是不会回滚的，这种情况下会造成自增列数据非连续增长。

以上所述都是各个MySql版本的默认实现，MySql 5.1引入了一个新的参数 innodb\_autoinc\_lock\_mode 通过修改这个字段的值，可以改变InnoDB生成自增列的策略，其值总结如下：

| **值** | **名称** | **含义** |
| --- | --- | --- |
| 0 | traditional lock mode | 每次插入语句执行都会全表加锁至语句结束，5.1版本之前默认实现 |
| 1 | consecutive lock mode | 简单插入不再全表加锁，INSERT ... SELECT类的语句才持有全表锁，5.1至8.0默认实现 |
| 2 | interleaved lock mode | INSERT ... SELECT类的语句也不会全表加锁，只有生成自增列数据时才加锁，8.0之后默认实现 |

不推荐显式指定自增列数据，因为在5.7以及之前的版本，如果通过update语句显式指定一个比SELECT MAX(\*ai\_col\*)还大的自增列值，后续insert语句可能会抛"Duplicate entry"错误，这一点在8.0版本之后也有了改变，如果通过显式的update语句显式指定一个比SELECT MAX(\*ai\_col\*)还大的自增列值，那该值就会被持久化，后续的自增列值都从该值开始生成。