超全面MySQL语句加锁分析（下篇）（求转）

原创 小孩子4919 我们都是小青蛙 2019-05-16

说在前面的话

本文是用来系统阐述在MySQL中，不同语句在各种条件下的加锁情况，并不是解释各种锁是什么（或者说加锁的本质是什么），大家如果不理解什么是 MVCC 、 ReadView 、 正经记录锁 、 gap锁 、 next-key锁 、 插入意向锁 这些概念的，可以参考 MySQL 的官方文档，或者直接参照《MySQL是怎样运行的：从根儿上理解MySQL》这本小册（里边有比官方文档更贴心，更详细的解释，文章中涉及到的所有概念均在小册中有详细解释，有疑惑，并且有兴趣的同学可以扫描下边二维码看看）：

建议：

1. 本篇文章不适合碎片化时间阅读，最好使用电脑观看，或者将字体调到最小效果好一些

2. 可能一下子看不完，关注 + 收藏 + 好看 + 转发一波

3. 不要跳着看

4. 一定要先看过上两篇文章：

- 超全面MySQL语句加锁分析（上篇）（求转）
- 超全面MySQL语句加锁分析（中篇）（求转）

INSERT语句

前边唠叨锁的细节时说过， INSERT 语句一般情况下不加锁，不过当前事务在插入一条记录前需要先定位到该记录在 B+树 中的位置，如果该位置的下一条记录已经被加了 gap锁 （ next-key锁 也

包含 gap锁 ，之后就不强调了），那么当前事务会在该记录上加上一种类型为 插入意向锁 的锁，并且事务进入等待状态。关于 插入意向锁 由于我们之前已经详细唠叨过了，就不多说了。

下边要看的是两种 INSERT 语句遇到的特殊情况：

遇到重复键（duplicate key）

在插入一条新记录时，首先要做的事情其实是定位到这条新记录应该插入到 B+树 的哪个位置。如果定位位置时发现了有已存在记录的主键或者唯一二级索引列与待插入记录的主键或者唯一二级索引列相同（不过可以有多条记录的唯一二级索引列的值同时为 NULL ），那么此时此时是会报错的。比方说我们插入新记录，该记录的主键值已经被包含在 hero 表中了：

```
mysql> BEGIN;
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO hero VALUES(20, 'g关羽', '蜀');
ERROR 1062 (23000): Duplicate entry '20' for key 'PRIMARY'
```

当然，在生成报错信息前，其实还需要做一件非常重要的事情 —— 对聚簇索引中number值为20的那条记录加S锁。不过具体的行锁类型在不同隔离级别下是不一样的：

- 在 READ UNCOMMITTED/READ COMMITTED 隔离级别下，加的是 S型正经记录锁 。
- 在 REPEATABLE READ/SERIALIZABLE 隔离级别下，加的是 S型next-key锁 。

如果是唯一二级索引列值重复，比方说我们再把普通二级索引 idx_name 改为唯一二级索引 uk_name ：

```
ALTER TABLE hero DROP INDEX idx_name, ADD UNIQUE KEY uk_name (name);
```

然后执行

```
mysql> BEGIN;
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO hero VALUES(30, 'c曹操', '魏');
ERROR 1062 (23000): Duplicate entry 'c曹操' for key 'uk_name'
```

很显然， hero 表中之前就包含 name 值为 'c曹操' 的记录，如果再插入一条 name 值为 'c曹操' 的新记录时，虽然插入对应的聚簇索引记录没问题，但是在插入 uk_name 唯一二级索引记录时便会报错，不过在报错之前还是会把 name 值为 'c曹操' 那条二级索引记录加一个 S锁 。需要注意的是，不管是哪个隔离级别，针对在插入新记录时遇到重复的唯一二级索引列的情况，会对已经在B+树中的唯一二级索引记录加next-key锁。

> 小贴士： 按理说在READ UNCOMMITTED/READ COMMITTED隔离级别下，不应该出现next-key锁，这主要是考虑到如果只加正经记录锁的话，在一些情况下可能出现有多条记录的唯一二级索引列都相同的情况。当然，出现这种情况的原因比较复杂，我们这里就不多说了。

另外，如果我们使用的是 INSERT ... ON DUPLICATE KEY ... 这样的语法来插入记录时，如果遇到主键或者唯一二级索引列值重复的情况，会对 B+树 中已存在的相同键值的记录加 X锁 ，而不是 S锁 。

外键检查

大家别忘了 MySQL 还是一个支持 外键 的数据库，比方说我们再为三国英雄的战马建一个表：

```
CREATE TABLE horse (
    number INT PRIMARY KEY,
    horse_name VARCHAR(100),
    FOREIGN KEY (number) REFERENCES hero(number)
)Engine=InnoDB CHARSET=utf8;
```

这样 hero 表就算是一个父表，新建的 horse 表就算一个子表，其中 horse 表的 number 列是参照 hero 表的 number 列。现在如果我们向子表中插入一条记录时：

- 如果待插入记录的 number 值在 hero 表中能找到。

  比方说我们为 horse 表中新插入的记录的 number 值为 8 ，而在 hero 表中 number 值为 8 的记录代表 曹操 ，他的马是 绝影 ：

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO horse VALUES(8, '绝影');
Query OK, 1 row affected (5 min 58.04 sec)
```

在插入成功之前，不论当前事务的隔离级别是什么，只需要直接给父表 hero 的 number 值为 3 的记录加一个 S型正经记录锁 。

- 如果待插入记录的 number 值在 hero 表中找不到。

  比方说我们为 horse 表中新插入的记录的 number 值为 2 ，而在 hero 表中不存在 number 值为 2 的记录：

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO horse VALUES(5, '绝影');
Query OK, 1 row affected (5 min 58.04 sec)
```

  虽然插入失败了，但是这个过程中需要对父表 hero 的某些记录进行加锁：

  - 在 READ UNCOMMITTED/READ COMMITTED 隔离级别下，并不对记录加锁。
  - 在 REPEATABLE READ/SERIALIZABLE 隔离级别下，加的是 gap锁 。

小结

MySQL的语句加锁分析暂时告一段落，但并没有把所有的情形都列举出来，只是给出了一个大致轮廓，希望各位在之后的工作学习中再多总结学些，如有疑问，可以联系小孩子哈～ 关注小青蛙，全都是技术干货哈：

阅读原文