MySQL学习笔记（Day038：purge死锁_backup_1）

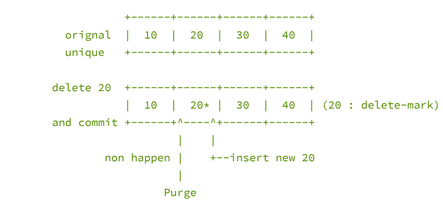
MySQL学习

一. Purge死锁举例

1.1 场景说明

a (auto_increment)	b (unique key)
1	10
2	20
3	30
4	40

表中存在记录（unique key）10，20，30，40（且有 自增主键 ），现在删除记录 20 ，并且已经 提交 了该事物。 purge 线程此时还 没有回收 该记录，且此时又 插入 新的记录 20 。




```
# 自增主键图中没有给出
```

> 回顾插入过程 **完整的插入过程如下：**
>
> 假设现在有记录 10，30，50，70 ；且为 unique key ，需要插入记录 25 。
>
> 1. 找到 小于等于25的记录 ，这里是 10
>   - 如果记录中已经 存在记录25 ，且带有 唯一性约束 ，则需要在 记录25 上增加 S Gap-lock （ **purge的案例中，老 记录20* 要加S lock的原因** ）
>   - 不直接报错退出或者提示已存在的原因，是因为有可能之前的 记录25 标记为删除(delete-mark)，然后等待 purge
>   - 如果 假设 这里 没有S Gap-Lock ，此时 记录30 上也 没有锁 的，按照下面的步骤，可以插入 两个25 ，这样就 破坏了唯一性约束
> 2. 找到 记录10的下一条记录 ，这里是 30
> 3. 判断 下一条记录30 上是否有锁（ *如果有=25的情况，后面再讨论* ）
>   - 判断 30 上面如果 没有锁 ，则 可以插入
>   - 判断 30 上面如果有 Record Lock ，则 可以插入
>   - 判断 30 上面如果有 Gap Lock / Next-Key Lock ，则无法插入，因为锁的范围是 (10, 30) / （10, 30]；在 30 上增加 insert intention lock（ *此时处于 waiting 状态* ），当 Gap Lock / Next-Key Lock 释放时，等待的事物（transaction）将被 唤醒 ，此时 记录30 上才能获得 insert intention lock ，然后再插入 记录25

在这个场景中，新插入的记录 20 ，和已经存在的记录 20* 相等，且带有唯一约束，那此时就需要在记录 20* 上增加 S lock（with gap）

1.2. 演示

因为要模拟插入记录 20* 的时候，老的 记录20 要存在，所以使用debug版本，将 purge线程停掉 。

```
[root@MyServer ~]> mysqld-debug  --version
mysqld-debug  Ver 5.7.11-debug for linux-glibc2.5 on x86_64 (MySQL Community Server - Debug (GPL))

[root@MyServer ~]> mysqld-debug  --datadir=/data/mysql_data/5.7.11/ &
[1] 1493
[root@MyServer ~]> netstat -tunlp | grep 3306
tcp        0      0 0.0.0.0:3306                0.0.0.0:*                   LISTEN      1493/mysqld-debug


--
-- 终端会话1
--
mysql> create table test_purge(a int auto_increment primary key, b int , unique key(b));
Query OK, 0 rows affected (0.20 sec)

mysql> insert into test_purge(b) values (10),(20),(30),(40);
Query OK, 4 rows affected (0.05 sec)

mysql> commit; -- autocommit=0 in my.cnf
Query OK, 0 rows affected (0.03 sec)

mysql> set global innodb_purge_stop_now=1;
-- show这个变量，结果还是off，这个不用管，purge线程已经停止了
Query OK, 0 rows affected (0.00 sec)

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

mysql> delete from test_purge where b=20;
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.02 sec)

--
-- 终端会话2
--
mysql> select * from test_purge;
+---+------+
| a | b    |  -- 20的那条记录已经删除，但是还没有被purge（purge线程停止）
+---+------+
| 1 |   10 |
| 3 |   30 |
| 4 |   40 |
+---+------+
3 rows in set (0.00 sec)

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

mysql> insert into test_purge(b) values(20);
Query OK, 1 row affected (0.04 sec)

--
-- 终端会话3
--
mysql> show engine innodb status\G
-- ---------------省略其他输出---------------
---TRANSACTION 9497, ACTIVE 19 sec
3 lock struct(s), heap size 1160, 3 row lock(s), undo log entries 1
MySQL thread id 3, OS thread handle 139922002294528, query id 26 localhost root cleaning up
TABLE LOCK table `burn_test`.`test_purge` trx id 9497 lock mode IX
RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9497 lock mode S  -- S lock （with gap）
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 32
-- heap no=3表示是第二个插入的记录20
-- 且info bits为32，表示记录被标记删除了
 0: len 4; hex 80000014; asc     ;;   -- 记录为20
 1: len 4; hex 80000002; asc     ;;   -- 对应的主键为2

Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
-- heap no=4表示的是20的下一个记录30
-- 且该记录上也有S lock
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;


RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9497 lock mode S locks gap before rec
Record lock, heap no 6 PHYSICAL RECORD: n_fields 2; compact format; info bits 0  -- heap no=6为新插入的记录20，从隐式锁提升为显示锁
 0: len 4; hex 80000014; asc     ;;
 1: len 4; hex 80000005; asc     ;;
```

1. 因为是唯一索引，需要做唯一性检查，从老的记录 20* 开始检查（ *第一个小于等于自己的值* ），则此时 20* 上要加上一把 S lock ，然后往下检查到第一个不相等的记录，即 记录30 ，然后退出，但是这个 记录30 也要 加上S lock
2. 在插入 新的记录20 的时候，发现下一条记录30上有锁，则自己插入的时的 隐式锁 提升为 显示锁 （ *见插入步骤* ）
3. 目前锁住的范围是 (10,20], (20,30]
4. 新插入的记录20本身是一把 S-Gap Lock （ *前面20*的有S lock了，由于是唯一索引，本身其实就不需要有记录锁了，有GAP就够了* ）

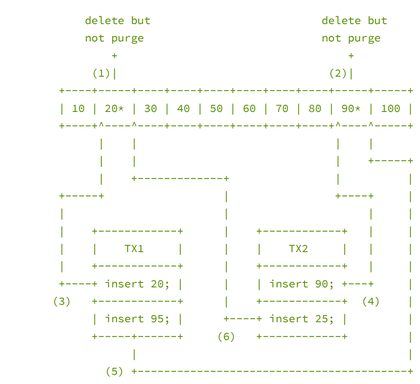
所以记录25无法插入（锁等待）

```
--
-- 终端会话1
--
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_purge(b) values(25);
ERROR 1205 (HY000): Unknown error 1205   -- 等待了一段时间后,超时

--
-- 终端会话3
--
mysql> show engine innodb status\G
-- ----------------省略其他输出----------------
---TRANSACTION 9508, ACTIVE 3 sec inserting
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1160, 1 row lock(s), undo log entries 1
MySQL thread id 5, OS thread handle 13992200560768, query id 46 localhost root update
insert into test_purge(b) values(25)  --  插入的25在等待
------- TRX HAS BEEN WAITING 3 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9508 lock_mode X locks gap before rec insert intention waiting
Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;

------------------
TABLE LOCK table `burn_test`.`test_purge` trx id 9508 lock mode IX
RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9508 lock_mode X locks gap before rec insert intention waiting
Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;

---TRANSACTION 9503, ACTIVE 10 sec
3 lock struct(s), heap size 1160, 3 row lock(s), undo log entries 1
MySQL thread id 7, OS thread handle 13992200202828288, query id 44 localhost root cleaning up
TABLE LOCK table `burn_test`.`test_purge` trx id 9503 lock mode IX
RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9503 lock mode S
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 32
 0: len 4; hex 80000014; asc     ;;
 1: len 4; hex 80000002; asc     ;;

Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;

RECORD LOCKS space id 47 page no 4 n bits 72 index b of table `burn_test`.`test_purge` trx id 9503 lock mode S locks gap before rec
Record lock, heap no 6 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 80000014; asc     ;;
 1: len 4; hex 80000007; asc     ;;
```

> 这个例子中出现了 锁等待 ，就要 警惕 了，如果有 两个事物相互等待 ，就是 死锁 了

Purge死锁演示

```
--
-- 终端会话1
--
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_purge(b) values(50),(60),(70),(80),(90),(100);
Query OK, 6 rows affected (0.05 sec)

mysql> commit ;
Query OK, 0 rows affected (0.03 sec)

--
-- 终端会话2
--
mysql> rollback; -- 回滚掉，不插入新的20，  老的记录20还在页里面
Query OK, 0 rows affected (0.00 sec)

--
-- 终端会话1
--
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from test_purge where b=90;  -- 删除90，但是数据还在页上
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

--
-- 终端会话2
--
mysql> select * from test_purge;
+----+------+
| a  | b    |
+----+------+
|  1 |   10 |
|  3 |   30 |
|  4 |   40 |
|  5 |   50 |
|  6 |   60 |
|  7 |   70 |
|  8 |   80 |
| 10 |  100 |
+----+------+
8 rows in set (0.00 sec)
```



> **步骤(1)：**删除20，且没有被purge
> **步骤(2)：**删除90，且没有被purge
> **步骤(3)：**TX1 插入20, 可以插入成功
> **步骤(4)：**TX2 插入90, 可以插入成功
> **步骤(5)：**TX1 插入95, 锁等待
> **步骤(6)：**TX2 插入25, 锁等待
>
> **步骤（5）和 步骤（6）最终因为相互等待，而导致有一个死锁**

```
--
-- 终端会话1 （步骤3）
--
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_purge(b) values(20);
Query OK, 1 row affected (0.00 sec)

--
-- 终端会话2 （步骤4）
--
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_purge(b) values(90);
Query OK, 1 row affected (0.04 sec)

--
-- 终端会话3
--
mysql> show engine innodb status\G
-- ----------------省略其他输出----------------
3 lock struct(s), heap size 1160, 3 row lock(s), undo log entries 1
MySQL thread id 10, OS thread handle 140173485033216, query id 74 localhost root cleaning up
TABLE LOCK table `burn_test`.`test_purge` trx id 10034 lock mode IX
RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10034 lock mode S  -- S Lock
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 32  -- heap no=3为删除的记录20，上面有S lock
 0: len 4; hex 80000014; asc     ;;
 1: len 4; hex 80000002; asc     ;;

Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0  -- 由于唯一性，被扫描到的记录30上，也有S lock
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;

RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10034 lock mode S locks gap before rec
Record lock, heap no 13 PHYSICAL RECORD: n_fields 2; compact format; info bits 0  -- 新插入的记录20，  上面有 S Gap lock
 0: len 4; hex 80000014; asc     ;;
 1: len 4; hex 8000000c; asc     ;;

---TRANSACTION 10029, ACTIVE 69 sec
3 lock struct(s), heap size 1160, 3 row lock(s), undo log entries 1
MySQL thread id 11, OS thread handle 140173484766976, query id 73 localhost root cleaning up
TABLE LOCK table `burn_test`.`test_purge` trx id 10029 lock mode IX
RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10029 lock mode S
Record lock, heap no 10 PHYSICAL RECORD: n_fields 2; compact format; info bits 32 -- 删除的记录90上 有一把 S lock
 0: len 4; hex 8000005a; asc    Z;;
 1: len 4; hex 80000009; asc     ;;

Record lock, heap no 11 PHYSICAL RECORD: n_fields 2; compact format; info bits 0  -- 由于唯一性，被扫描到的记录100上有一把S lock
 0: len 4; hex 80000064; asc    d;;
 1: len 4; hex 8000000a; asc     ;;

RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10029 lock mode S locks gap before rec
Record lock, heap no 12 PHYSICAL RECORD: n_fields 2; compact format; info bits 0 -- 新插入的记录90，  上面有 S Gap  lock
 0: len 4; hex 8000005a; asc    Z;;
 1: len 4; hex 8000000b; asc     ;;
```

> **此时的锁的范围：**
> 事物1 ： S Lock (10, 20], (20, 30]
> 事物2 ： S Lock (80, 90], (90, 100]
>
> 事物1 插入 95 ，需要 等待事物2 释放
> 事物2 插入 25 ，需要 等待事物1 释放
>
> 又是典型的 AB-BA死锁

```
--
-- 终端会话1 （步骤5）
--
mysql> insert into test_purge(b) values(95);
Query OK, 1 row affected (0.64 sec)

--
-- 终端会话2 （步骤6）
--
mysql> insert into test_purge(b) values(25);
ERROR 1213 (40001): Unknown error 1213     -- Error 1213， dead lock后，被回滚了

--
-- 终端会话3
--
mysql> show engine innodb status\G
-- ---------------省略其他输出----------------
------------------------
LATEST DETECTED DEADLOCK
------------------------
2016-03-04 17:55:38 0x7f7caec43700
*** (1) TRANSACTION:
TRANSACTION 10035, ACTIVE 41 sec inserting
mysql tables in use 1, locked 1
LOCK WAIT 4 lock struct(s), heap size 1160, 4 row lock(s), undo log entries 2
MySQL thread id 13, OS thread handle 140173485033216, query id 100 localhost root update
insert into test_purge(b) values(95)  -- 事物1 想要插入95，在等待100的插入意向锁
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10035 lock_mode X locks gap before rec insert intention waiting
Record lock, heap no 11 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 80000064; asc    d;;
 1: len 4; hex 8000000a; asc     ;;

*** (2) TRANSACTION:
TRANSACTION 10040, ACTIVE 24 sec inserting, thread declared inside InnoDB 5000
mysql tables in use 1, locked 1
4 lock struct(s), heap size 1160, 4 row lock(s), undo log entries 2
MySQL thread id 14, OS thread handle 140173484766976, query id 101 localhost root update
insert into test_purge(b) values(25)  -- 事物2要插入 记录 25
*** (2) HOLDS THE LOCK(S):   -- 事物2 持有 90和100的S lock （with gap）
RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10040 lock mode S
Record lock, heap no 10 PHYSICAL RECORD: n_fields 2; compact format; info bits 32
 0: len 4; hex 8000005a; asc    Z;;
 1: len 4; hex 80000009; asc     ;;

Record lock, heap no 11 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 80000064; asc    d;;
 1: len 4; hex 8000000a; asc     ;;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:   -- 事物2 在等待 30 这把插入意向锁（被事物1占着）
RECORD LOCKS space id 49 page no 4 n bits 80 index b of table `burn_test`.`test_purge` trx id 10040 lock_mode X locks gap before rec insert intention waiting
Record lock, heap no 4 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 8000001e; asc     ;;
 1: len 4; hex 80000003; asc     ;;

*** WE ROLL BACK TRANSACTION (2)  -- 最终回滚到了事物2
```

> 上述演示中，并 没有重复 的值，但还是出现了死锁，原因就是使用了 唯一索引 ，导致检查唯一性时，产生了 S lock

- **生该问题的场景：**
  1. 使用了唯一索引（主要原因）
  2. 大量delete数据后，又立即插入了数据
  3. 插入的数据和部分删除的数据的唯一索引一样，且purge还没有来得及回收删除的数据
- **解决办法**
  1. delete后等待较长时间后（增大purge线程数，等Purge回收），再插入新数据（不推荐）
  2. 使用普通索引
  3. 仍然使用唯一索引，但是插入前要对 唯一索引进行排序（只会有等待，不会有死锁）

> **Tips**
>
> 1. 老师视频演示的第一个 `create table zz (a int primary key);`,要能产生 S lock ，要使用 RR 的隔离级别； RC 隔离级别只能产生 S lock but not gap
> 2. 由于这个例子是的特殊性（*原地更新*），无法演示出死锁，所以采用上述的例子演示
>   原地更新是指 这条记录 在事物过程 中，是否发生原地更新的行为，而不是事物提交后发生的更新（都是在事物中）

二. 备份（一）

2.1. 备份的类型

以下三种方式都是 全量备份 的方式

1. **热备**（Hot Backup）
   - 在线备份
   - 对应用无影响（应用程序不会被阻塞*（其实有，只是时间很短）*，可以正常的读写，但是性能上还是有影响的）
2. **冷备**（Cold Backup）
   - 备份数据文件，最可靠的备份
   - 需要停机（最大的弊端）
   - 备份datadir下的所有文件
3. **温备**（Warm Backup）
   - 在线备份
   - 对应用影响很大
   - 通常加一个读锁（读不受影响，写被阻塞）

2.2. 备份工具

1. **ibbackup**
   - 官方备份工具
   - 收费
   - 物理备份
2. **xtrabackup**
   - 开源社区备份工具（*必须使用最新版本，否则备份出来的数据可能有问题*）
   - 开源免费
   - 物理备份
   - Xtrabackup老版本的问题
3. **mysqldump**
   - 官方自带备份工具，是可靠的，且 备份文件相对较小
   - 开源免费
   - 逻辑备份
     - 恢复速度较慢（需要重建索引等等）

> 注意:
>
> 1. 有的热备都只能使InnoDB存储引擎表
> 2. 其他存储引擎表只能是温备

- **物理备份**
  - 备份了 表空间 的数据，和冷备类似
- **逻辑备份**
  - 备份了 表中 的数据（导出的是一条条SQL）

	逻辑备份	物理备份
备份方式	备份数据库逻辑内容	备份数据库物理文件
优点	备份文件相对较小 （只备份表中的数据与结构）	恢复速度比较快 （物理文件恢复基本已经完成恢复）
缺点	恢复速度较慢 （需要重建索引、存储过程等）	备份文件相对较大 （备份表空间，包含数据与索引）
对业务影响	缓冲池污染、I/O负载加大	I/O负载加大
代表工具	mysqldump	ibbackup、xtrabackup

- 从以下维度选择备份方式：
  - 备份速度
  - 恢复速度
  - 备份大小
  - 对业务影响