MySQL学习笔记（Day032：锁_5）

MySQL学习

一. 死锁举例（一）– 购物车

1.1. 环境说明

- 用户1 的购物车：
  1. product_id= 100 的商品
  2. product_id= 110 的商品
  3. product_id= 10 的商品
- 用户2 的购物车：
  1. product_id= 10 的商品
  2. product_id= 110 的商品
  3. product_id= 100 的商品

1. 当一个商品被购买时，其中有一个环节是 库存减1 ，即做update操作（set stock=stock-1 where product_id=xx）
2. 当两个用户购买的产品的 product_id 顺序交叉，且两个用户同时下订单时，购物车中的每个商品都会执行 库存减1 的操作，即 每个用户 在各自的 一个事物 中做 三次update 操作，此时即发生了 死锁 （类似AB-BA死锁）

1.2. 解决办法

在应用程序端进行修改，即提交购物车订单时，对 订单进行排序 ，然后再执行提交。这样只会出现锁等待，而不会出现死锁

二. lock in share mode

```
mysql> select version();
+------------+
| version()  |
+------------+
| 5.6.27-log | -- 在MySQL5.6中可以演示出来，MySQL5.7版本无法得到锁信息
+------------+
1 row in set (0.00 sec)


--
-- 终端会话1
--
mysql> create table t_lock_6 (a int ,key(a));  -- 二级索引
Query OK, 0 rows affected (0.14 sec)

mysql> insert into t_lock_6 values(1),(10);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> set tx_isolation='REPEATABLE-READ';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t_lock_6;
+------+
| a    |
+------+
|    1 |
|   10 |
+------+
2 rows in set (0.00 sec)

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

mysql> select * from t_lock_6 where a = 8 lock in share mode;
Empty set (0.00 sec)

--
-- 终端会话2
--
mysql> set tx_isolation='REPEATABLE-READ';
Query OK, 0 rows affected (0.00 sec)

mysql> show engine innodb status\G
--  ----------省略部分输出----------
2 lock struct(s), heap size 360, 1 row lock(s)
MySQL thread id 1, OS thread handle 0x7f01d7e82700, query id 14 localhost root cleaning up
TABLE LOCK table `burn_test`.`t_lock_6` trx id 4880 lock mode IS
RECORD LOCKS space id 14 page no 4 n bits 72 index `a` of table `burn_test`.`t_lock_6` trx id 4880 lock mode S locks gap before rec  -- 在记录10上的一个Gap锁
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000000301; asc       ;;
```

> 即 select * from t_lock_6 where a = 8 lock in share mode; 这句话，让innodb在 记录10 上 产生 了一个 Gap Lock ，即锁住的范围是 (1, 10) （*因为是二级索引，所以是开区间*），这样一来 其他事物 就无法在这个范围内进行插入操作了

```
--
-- 终端会话2
--
mysql> insert into t_lock_6 select 5;  -- 无法插入
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql> insert into t_lock_6 select 8;  -- 无法插入
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction


--
-- 终端会话1
--
-- 但是此时 终端会话1 中的事物还没有提交，自己是可以插入的
-- 回到 终端会话1 的事物中

mysql> insert into t_lock_6 values(8);
Query OK, 1 row affected (0.00 sec)

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

mysql> select * from t_lock_6 ;
+------+
| a    |
+------+
|    1 |
|    8 |
|   10 |
+------+
3 rows in set (0.00 sec)
```

> 使用 lock in share mode 可以产生一个 Gap Lock ， 防止其他事物在这个Gap范围内插入数据 ；
> 同时如果 返回 的结果集 为空 ，则表示 查询条件不存在 ，相当于做了一次 唯一性检查 ；
> 此时（原事物）再进行插入对应条件的记录，可以确保 插入的记录 的 唯一性 ；
> 实际中如果要保证唯一性，可以设置为主键或者唯一索引

三. 死锁举例（二）

```
--
-- 终端会话1
--
mysql> set innodb_lock_wait_timeout=60;
Query OK, 0 rows affected (0.00 sec)

mysql> set tx_isolation="READ-COMMITTED";
Query OK, 0 rows affected (0.00 sec)

mysql> create table t_lock_8 (a int not null, unique key(a));
Query OK, 0 rows affected (0.14 sec)

mysql> insert into t_lock_8 values(1),(10);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from t_lock_8;
+----+
| a  |
+----+
|  1 |
| 10 |
+----+
2 rows in set (0.00 sec)

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

mysql> insert into t_lock_8 values(8);
Query OK, 1 row affected (0.00 sec)


--
-- 终端会话2
--
mysql> set tx_isolation="READ-COMMITTED";
Query OK, 0 rows affected (0.00 sec)

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

mysql> insert into t_lock_8 values(8);
-- waiting......



--
-- 终端会话3
--
mysql> set tx_isolation="READ-COMMITTED";
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t_lock_8 values(8);
-- waiting.....



mysql> show engine innodb status\G
--  ----------省略部分输出----------
------------
TRANSACTIONS
------------
Trx id counter 27209
Purge done for trx's n:o < 27206 undo n:o < 0 state: running but idle
History list length 322
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 422105834340864, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 27208, ACTIVE 1 sec inserting   --  终端会话3 中的事物
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 12, OS thread handle 140629332293376, query id 478 localhost root update
insert into t_lock_8 values(8)
------- TRX HAS BEEN WAITING 1 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27208 lock mode S locks rec but not gap waiting  -- 等待对 记录8 的S Lock的授权
Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a46; asc     jF;;
 2: len 7; hex c0000000390110; asc     9  ;;

------------------
TABLE LOCK table `burn_test`.`t_lock_8` trx id 27208 lock mode IX
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27208 lock mode S locks rec but not gap waiting
Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a46; asc     jF;;
 2: len 7; hex c0000000390110; asc     9  ;;

---TRANSACTION 27207, ACTIVE 2 sec inserting   --  终端会话2 中的事物
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 11, OS thread handle 140629332027136, query id 477 localhost root update
insert into t_lock_8 values(8)
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock mode S locks rec but not gap waiting  -- 等待对 记录8 的S Lock的授权
Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a46; asc     jF;;
 2: len 7; hex c0000000390110; asc     9  ;;

------------------
TABLE LOCK table `burn_test`.`t_lock_8` trx id 27207 lock mode IX
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock mode S locks rec but not gap waiting
Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a46; asc     jF;;
 2: len 7; hex c0000000390110; asc     9  ;;

---TRANSACTION 27206, ACTIVE 15 sec  --  终端会话1 中的事物
2 lock struct(s), heap size 1136, 1 row lock(s), undo log entries 1
MySQL thread id 13, OS thread handle 140629331760896, query id 475 localhost root cleaning up
TABLE LOCK table `burn_test`.`t_lock_8` trx id 27206 lock mode IX
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27206 lock_mode X locks rec but not gap  -- 对插入的 记录8 持有一个Record Lock
Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a46; asc     jF;;
 2: len 7; hex c0000000390110; asc     9  ;;
```

终端会话1 中的事物持有 记录8 的 Record Lock (X-Lock not gap)
终端会话2 中的事物在 等待 对 记录8 的 S-Lock 的授权
终端会话3 中的事物在 等待 对 记录8 的 S-Lock 的授权

```
--
-- 终端会话1
--
mysql> rollback;
Query OK, 0 rows affected (0.01 sec)



--
-- 终端会话2
--
mysql> insert into t_lock_8 values(8);
-- waiting.....   终端会话1 中的事物回滚后，这里显示插入成功
Query OK, 1 row affected (9.78 sec)



--
-- 终端会话3
--
mysql> insert into t_lock_8 values(8);
-- waiting.....   终端会话1 中的事物回滚后，这里显示  死锁
ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction
```

终端会话1 中的事物进行了回滚
终端会话2 中的事物插入记录8成功
终端会话3 中的事物则死锁

```
mysql> show engine innodb status\G
--  ----------省略部分输出----------
------------------------
LATEST DETECTED DEADLOCKLock
------------------------
2016-02-13 20:47:37 0x7fe6d1657700
*** (1) TRANSACTION:
TRANSACTION 27207, ACTIVE 15 sec inserting  --  终端会话2 中的事物
mysql tables in use 1, locked 1
LOCK WAIT 4 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 11, OS thread handle 140629332027136, query id 477 localhost root update
insert into t_lock_8 values(8)
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock_mode X locks gap before rec insert intention waiting  -- 终端会话2 中的事物在 等待 记录10 上的 插入意向锁（Gap Lock）
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000006a3a; asc     j:;;
 2: len 7; hex b800000049011d; asc     I  ;;

*** (2) TRANSACTION:
TRANSACTION 27208, ACTIVE 14 sec inserting, thread declared inside InnoDB 1   --  终端会话3 中的事物
mysql tables in use 1, locked 1
4 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 12, OS thread handle 140629332293376, query id 478 localhost root update
insert into t_lock_8 values(8)
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27208 lock mode S locks gap before rec  -- 终端会话3 中的 事物 持有 记录10 的 S-Gap Lock
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000006a3a; asc     j:;;
 2: len 7; hex b800000049011d; asc     I  ;;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27208 lock_mode X locks gap before rec insert intention waiting  --  终端会话3 中的事物在 等待 记录10 上的 插入意向锁（Gap Lock）
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000006a3a; asc     j:;;
 2: len 7; hex b800000049011d; asc     I  ;;

*** WE ROLL BACK TRANSACTION (2)  -- 回滚了 终端会话3 中的事物，所以 终端会话2 中才插入成功
------------
TRANSACTIONS
------------
Trx id counter 27213
Purge done for trx's n:o < 27213 undo n:o < 0 state: running but idle
History list length 324
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 422105834340864, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 422105834339952, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 422105834339040, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 27207, ACTIVE 19 sec
4 lock struct(s), heap size 1136, 3 row lock(s), undo log entries 1
MySQL thread id 11, OS thread handle 140629332027136, query id 477 localhost root cleaning up
TABLE LOCK table `burn_test`.`t_lock_8` trx id 27207 lock mode IX
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock mode S locks rec but not gap
RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock mode S locks gap before rec
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000006a3a; asc     j:;;
 2: len 7; hex b800000049011d; asc     I  ;;

Record lock, heap no 4 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000008; asc     ;;
 1: len 6; hex 000000006a47; asc     jG;;
 2: len 7; hex c1000000510110; asc     Q  ;;

RECORD LOCKS space id 153 page no 3 n bits 72 index a of table `burn_test`.`t_lock_8` trx id 27207 lock_mode X locks gap before rec insert intention
Record lock, heap no 3 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 8000000a; asc     ;;
 1: len 6; hex 000000006a3a; asc     j:;;
 2: len 7; hex b800000049011d; asc     I  ;;
```

> **死锁分析：**
>
> 原来加在 记录8 上的锁，都转移到了 记录10 上了(锁继承)
>
> 1. 因为 终端会话1 中的事物 回滚 了，则 记录8 标记为删除 (delete-mark)
> 2. 而 终端会话2 和 终端会话3 中的 事物 都对 记录8 添加了 S-Lock （*后台 Purge 线程会把 记录8 给删除*）
> 3. 此时下一个记录，即 记录10 会 继承 之前 记录8 上的锁，所以上面看到了 锁 的信息都在 记录10 上了
>
> 则此时的锁的状态是：
>
> 1. 终端会话2 持有记录10的 S Gap-Lock ，并且 等待 X locks gap insert intention lock
> 2. 终端会话3 持有记录10的 S Gap-Lock ，并且 等待 X locks gap insert intention lock
>
> ***X Lock 和 S Lock 是不兼容的***
>
> 终端会话3 的 insert intention lock 等待 终端会话2 的 S Gap-Lock 的释放；
> 终端会话2 的 insert intention lock 等待 终端会话3 的 S Gap-Lock 的释放；
>
> 所以产生了死锁（*其实还是AB-BA死锁*）

至于为什么加 S Gap-Lock ，是因为在插入之前（*Day30-1.1.插入的过程*）还需要多一步检查：如果记录中有 唯一约束 ，判断存在一条记录等于当前插入的记录时，则需要在这个记录加上 S Gap-Lock 。

完整的插入过程如下：

假设现在有记录 10，30，50，70 ；且为 主键 ，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
   - 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

对于死锁的官方例子中提及的第二个例子，线上是无法完全避免的，解决的办法就是 程序端重试 。

后续涉及Purge时，可以只使用 两条SQL 语句，就出现死锁现象，这里使用了3条SQL语句。

mysqld-debug 版本可以操作更多的参数（*比如关闭purge*），并且可以看到丰富的信息，可以用于故障诊断。