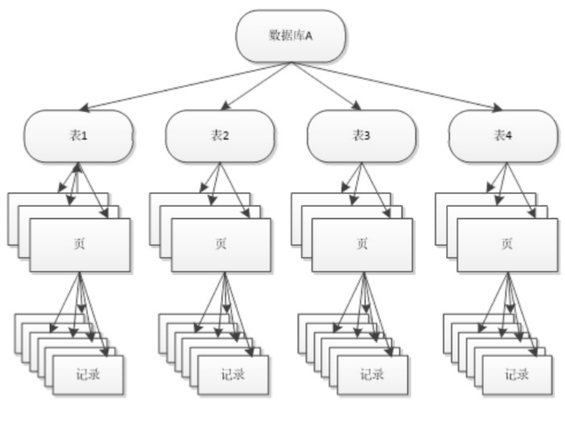
MySQL学习笔记（Day029：锁_2）

MySQL学习

一. 锁（二）

1.1. 意向锁介绍

1. 揭示下一层级请求的锁的类型
2. IS：事物想要获得一张表中某几行的共享锁
3. IX：事物想要获得一张表中某几行的排他锁
4. InnoDB存储引擎中意向锁都是 表锁



假如此时有 事物tx1 需要在 记录A 上进行加 X锁 ：

1. 在该记录所在的 数据库 上加一把 意向锁IX
2. 在该记录所在的 表 上加一把 意向锁IX
3. 在该记录所在的 页 上加一把 意向锁IX
4. 最后在该 记录A 上加上一把 X锁

假如此时有 事物tx2 需要对 记录B （*假设和记录A在同一个页中*）加 S锁 ：

1. 在该记录所在的 数据库 上加一把 意向锁IS
2. 在该记录所在的 表 上加一把 意向锁IS
3. 在该记录所在的 页 上加一把 意向锁IS
4. 最后在该 记录B 上加上一把 S锁

> 加锁是 从上往下 ， 一层一层 进行加的

锁兼容	X	IX	S	IS
X	冲突	冲突	冲突	冲突
IX	冲突	兼容	冲突	兼容
S	冲突	冲突	兼容	兼容
IS	冲突	兼容	兼容	兼容

> 意向锁都是 相互兼容 的，因为意向锁表示的是 下一层 在请求什么类型的锁

假如此时有 事物tx3 需要在 记录A 上进行加 S锁 ：

1. 在该记录所在的 数据库 上加一把 意向锁IS
2. 在该记录所在的 表 上加一把 意向锁IS
3. 在该记录所在的 页 上加一把 意向锁IS
4. 发现该记录被锁定（ tx1的X锁 ），那么 tx3需要等待 ，直到 tx1 进行commit

1.2. 意向锁的作用

- 意向锁 是为了实现 多粒度的锁 ，表示在数据库中不但能实现 行级别的锁 ，还可以实现 页级别的锁 ， 表级别的锁 以及 数据库级别的锁
- 如果没有意向锁，当你去锁一张表的时候，你就需要对表下的所有记录都进行加锁操作，且对其他事物刚刚插入的记录（游标已经扫过的范围）就没法在上面加锁了，此时就没有实现锁表的功能。

> 上述锁的操作都是在 内存 中，不会放在数据库中。且大部分加的都是意向锁，都是兼容的。
>
> 释放操作则是从记录锁开始 从下往上 进行释放

InnoDB 没有 数据库级别的锁，也 没有 页级别的锁（InnoDB只能在 表 和 记录 上加锁），所以InnoDB的 意向锁 只能加在 表 上，即InnoDB存储引擎中意向锁都是 表锁

1.3. 加锁以及查看

```
-- 终端1
mysql> desc t5;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | YES  |     | NULL    |       |
| b     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> select * from t5;
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
+------+------+
1 row in set (0.00 sec)

mysql> begin;  -- 开始一个事物
Query OK, 0 rows affected (0.01 sec)

mysql> select * from t5 where a=2 for update;  -- 加上一个排他锁
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
+------+------+
1 row in set (0.00 sec)


-- 终端2
mysql> show engine innodb status\G
-- ----------省略其他输出-------
---TRANSACTION 23076, ACTIVE 96 sec
2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 3, OS thread handle 139787704108800, query id 44 localhost root cleaning up
TABLE LOCK table `burn_test`.`t5` trx id 23076 lock mode IX -- 在表上加上了意向锁IX（TABLE LOCK）
RECORD LOCKS space id 62 page no 3 n bits 72 index GEN_CLUST_INDEX of table `burn_test`.`t5` trx id 23076 lock_mode X locks rec but not gap -- X locks rec but not gap 就是记录锁 （RECORD LOCK）
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
 0: len 6; hex 000000000625; asc      %;;
 1: len 6; hex 000000001390; asc       ;;
 2: len 7; hex 610000002c0d5f; asc a    , _;;
 3: len 4; hex 80000002; asc     ;;
 4: len 4; hex 80000002; asc     ;;

-- Record lock : 表示是锁住的记录
-- heap no 2 PHYSICAL RECORD: n_fields 5 : 表示锁住记录的heap no 为2的物理记录，由5个列组成
-- compact format : 表示这条记录存储的格式（Dynamic其实也是compact的格式）
-- info bits : 0 -- 表示这条记录没有被删除; 非0 -- 表示被修改或者被删除（32）

-- 输出上述信息的前提是 innodb_status_output_locks = 1
-- 可在配置文件中设置打开，不会影响运行时的性能
-- 只有在show engine innodb status时才会使用


-- 终端1
mysql> commit; -- 提交事物
Query OK, 0 rows affected (0.00 sec) -- 提交后就释放了锁，同时在终端2上就看不到锁的信息了

-- 注意如果直接select * from t5 where a=2 for update;终端2是看不到的，因为mysql默认会自动提交事物
```

- **INNODB_TRX**

```
-- 终端1
mysql> begin;  -- 开启一个事物
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t5 where a=2 for update;  -- 继续锁住该记录
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
+------+------+
1 row in set (0.00 sec)

-- 终端2

mysql> use information_schema;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select * from INNODB_TRX\G
*************************** 1. row ***************************
                    trx_id: 23077
                 trx_state: RUNNING
               trx_started: 2016-01-24 12:42:44
     trx_requested_lock_id: NULL
          trx_wait_started: NULL
                trx_weight: 2
       trx_mysql_thread_id: 13
                 trx_query: NULL -- 事物运行的SQL语句，这里是NULL
                                 -- 因为他是指当前运行的SQL语句
                                 -- 如果运行完了，就看不到了
       trx_operation_state: NULL
         trx_tables_in_use: 0
         trx_tables_locked: 1    -- 表锁IX
          trx_lock_structs: 2    -- 总共有2把锁
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 1    -- 记录锁
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: READ COMMITTED  -- 当前数据库隔离级别是RC
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 0
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

- **INNODB_LOCKS和INNODB_LOCK_WAITS**

```
-- 终端2
-- 下面两个表是要一起对比查看才有意义的，当前没有两个事物（仅终端1中一个事物）进行相互阻塞和等待，所以目前两个表是空的
mysql> select * from INNODB_LOCKS;
Empty set (0.00 sec)

mysql> select * from INNODB_LOCK_WAITS;
Empty set (0.00 sec)


-- 所以为了演示，需要再开一个会话 终端3
-- 终端3
mysql> set innodb_lock_wait_timeout=60; -- 设置锁等待时间为60秒，方便看到锁的信息，线上根据实际情况自行修改
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t5 where a=2  lock in share mode;  -- 这里如果超时时间设置短，可能来不及看
                                                        -- 可通过innodb_lock_wait_timeout进行设置
-- 此时hang住在这里，进行等待，因为终端1中 for update 还没有commit


-- 终端2
-- 回到终端2中，查看之前两个表的信息
mysql> select * from INNODB_LOCKS\G
*************************** 1. row ***************************
    lock_id: 23078:62:3:2
lock_trx_id: 23078  -- 锁ID为23078
  lock_mode: S      -- S 锁
  lock_type: RECORD
 lock_table: `burn_test`.`t5`
 lock_index: GEN_CLUST_INDEX
 lock_space: 62  -- space id
  lock_page: 3   -- page number
   lock_rec: 2   -- heap number
  lock_data: 0x000000000625
*************************** 2. row ***************************
    lock_id: 23077:62:3:2
lock_trx_id: 23077   -- 锁ID为23077
  lock_mode: X       -- X 锁（for  update）
  lock_type: RECORD
 lock_table: `burn_test`.`t5`
 lock_index: GEN_CLUST_INDEX
 lock_space: 62
  lock_page: 3
   lock_rec: 2
  lock_data: 0x000000000625
2 rows in set (0.00 sec)

mysql> select * from INNODB_LOCK_WAITS\G
*************************** 1. row ***************************
requesting_trx_id: 23078  -- 请求的事物ID （S锁），终端3
requested_lock_id: 23078:62:3:2  -- 由txid，space，page_no，heap_no组成
  blocking_trx_id: 23077  -- 阻塞上面请求的事物ID（X锁），终端1
 blocking_lock_id: 23077:62:3:2
1 row in set (0.00 sec)
```

- MySQL5.6 下查看锁信息

```
SELECT
    r.trx_id waiting_trx_id,
    r.trx_mysql_thread_id waiting_thread,
    r.trx_query waiting_query,
    b.trx_id blocking_trx_id,
    b.trx_mysql_thread_id blocking_thread,
    b.trx_query blocking_query
FROM
    information_schema.innodb_lock_waits w
        INNER JOIN
    information_schema.innodb_trx b ON b.trx_id = w.blocking_trx_id
        INNER JOIN
    information_schema.innodb_trx r ON r.trx_id = w.requesting_trx_id;

-- 在mysql中执行上述SQL，得到以下结果：（保证第二个SQL不超时，才能看到）
*************************** 1. row ***************************
 waiting_trx_id: 23078  -- 请求的事物ID
 waiting_thread: 21     -- 请求的线程ID
  waiting_query: select * from t5 where a=2  lock in share mode  -- 等待的SQL语句
blocking_trx_id: 23077  -- 阻塞上面请求的事物的ID
blocking_thread: 22     -- 阻塞的线程ID
 blocking_query: NULL   -- 阻塞的当前的SQL，这个是无法看到的，除非SQL还没有执行完成（不一定是该事物当中的最后一条SQL语句）
1 row in set (0.00 sec)
```

- MySQL 5.7 下查看锁信息

```
-- 继续执行 终端1/终端3 内的SQL语句，使其中有一个线程发生阻塞

mysql> use sys
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed


mysql> select * from innodb_lock_waits\G
*************************** 1. row ***************************
                wait_started: 2016-01-25 21:00:04  -- 开始的时间
                    wait_age: 00:00:04   -- 等待的时间
               wait_age_secs: 4          -- 等待秒数
                locked_table: `burn_test`.`t5`   -- 锁住的表（意向锁）
                locked_index: GEN_CLUST_INDEX    -- 锁住的是系统生成的聚集索引，锁都是在索引上的
                 locked_type: RECORD             -- 锁的类型，记录锁
             waiting_trx_id: 421206427405024    -- 等待的事物ID
        waiting_trx_started: 2016-01-25 21:00:04
            waiting_trx_age: 00:00:04
    waiting_trx_rows_locked: 1
  waiting_trx_rows_modified: 0
                waiting_pid: 6
              waiting_query: select * from t5 where a=2  lock in share mode
            waiting_lock_id: 421206427405024:62:3:2    -- 事物ID：space：page_No：heap_no
          waiting_lock_mode: S    --  等待（请求）的锁的类型
            blocking_trx_id: 23592
               blocking_pid: 7
             blocking_query: NULL
           blocking_lock_id: 23592:62:3:2
         blocking_lock_mode: X    -- 阻塞的锁的类型
       blocking_trx_started: 2016-01-25 20:59:48
           blocking_trx_age: 00:00:20
   blocking_trx_rows_locked: 2
 blocking_trx_rows_modified: 0
    sql_kill_blocking_query: KILL QUERY 7  -- 给出了建议
sql_kill_blocking_connection: KILL 7
1 row in set (0.01 sec)
```

> MySQL 5.6 通过导入sys库，也可以支持该视图
>
> **锁都是 锁在索引 上的，无论是主键还是二级索引，通过 `locked_index` 进行查看**
>
> 当超过 `innodb_lock_wait_timeout` 设置的阈值，等待（请求）的事物就会报锁超时。在某些业务场景下，锁超时是无法避免的。

二. 锁与并发

- locking （ 锁 ）
- concurrency control （ 并发控制 ）
- isolation （ 隔离级别 ）
- serializability （ 序列化 ）

以上四个其实在数据库中讲的是同一个概念； 锁 是用来实现 并发控制 ， 并发控制 用来实现 隔离级别 ， 隔离级别 是通过 锁 来控制的， 锁 的目的为了使得事物之间的执行是 序列化 的

2.1. 事物隔离级别

1. READ UNCOMMITED
2. `READ COMMITTED`
    - Oracle、DB2、Microsoft SQL Server（默认）
    - 解决 脏读 （ANSI SQL）
3. `REPEATABLE READ`
    - InnoDB（默认）
    - 解决 脏读、不可重复读 （ANSI SQL）
    - **InnoDB中的RR解决了幻读问题 （实现了事物的隔离级别）**
4. SERIALIZABLE
    - 解决 脏读、不可重复读 和 幻读 （ANSI SQL）

> 隔离级别 越低，事物请求的锁 越少 或者保持锁的时间就 越短。
>
> - **什么是隔离性?**
>
> 一个事物 **所做的** 修改 **，对** 其他事物 **是** 不可见 **的，好似是** 串行 **执行的**

```
-- 终端1
mysql> create table test_rc (a int, b int , c int);
Query OK, 0 rows affected (0.14 sec)

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

mysql> insert into test_rc values(1,2,3);
Query OK, 1 row affected (0.00 sec)

mysql> select * from test_rc;
+------+------+------+
| a    | b    | c    |
+------+------+------+
|    1 |    2 |    3 |
+------+------+------+
1 row in set (0.00 sec)


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

mysql> insert into test_rc values(4,5,6);
Query OK, 1 row affected (0.00 sec)

mysql> commit;  -- 事物提交
Query OK, 0 rows affected (0.03 sec)

-- 终端1
mysql> select * from test_rc;
+------+------+------+
| a    | b    | c    |
+------+------+------+
|    1 |    2 |    3 |
|    4 |    5 |    6 |  -- 一个事物的修改出现在了另外一个事物中，其实是不符合隔离性的要求的
+------+------+------+
2 rows in set (0.00 sec)

mysql> select @@tx_isolation;  -- 查看当前事物的隔离级别
+----------------+
| @@tx_isolation |
+----------------+
| READ-COMMITTED |  -- 当前是RC的，也就是Oracle，MSSQL 的默认隔离级别，其实是不复合隔离性的要求的
+----------------+
1 row in set (0.00 sec)
```

> **线上环境一般使用 `READ-COMMITTED`**