链滴

mysql 锁

作者：AshShawn

原文链接：https://ld246.com/article/1648134929228

来源网站：链滴

1. 全局锁

1.1 FTWRL

对整个数据库加锁

Flush tables with read lock (FTWRL) //整库只读

数据更新语句（数据的增删改）、数据定义语句（包括建表、修改表结构等）和更新类事务的提交语都会被阻塞。

全局锁的典型使用场景是，做全库逻辑备份

1.2 single-transaction 一致性读

使用msqldump工具并加上–single-transaction可以开启一个事务,进行逻辑备份,数据也是可以正常新的

set global readonly=true //全库只读,readonly一般用来判断主库还是备库,遇到异常时FTWRL会释全局锁,readonly遇到异常会一直导致库不可写,风险较高

2. 表级锁

MySQL 里面表级别的锁有两种：一种是**表锁**，一种是**元数据锁**（meta data lock，MDL)。

2.1 表锁:

lock tables ... read/write
unlock tables

如果在某个线程 A 中执行 lock tables t1 read, t2 write; 这个语句，则其他线程写 t1、读写 t2 的语都会被阻塞。同时，线程 A 在执行 unlock tables 之前，也只能执行读 t1、读写 t2 的操作。连写 t1 都不允许，自然也不能访问其他表。

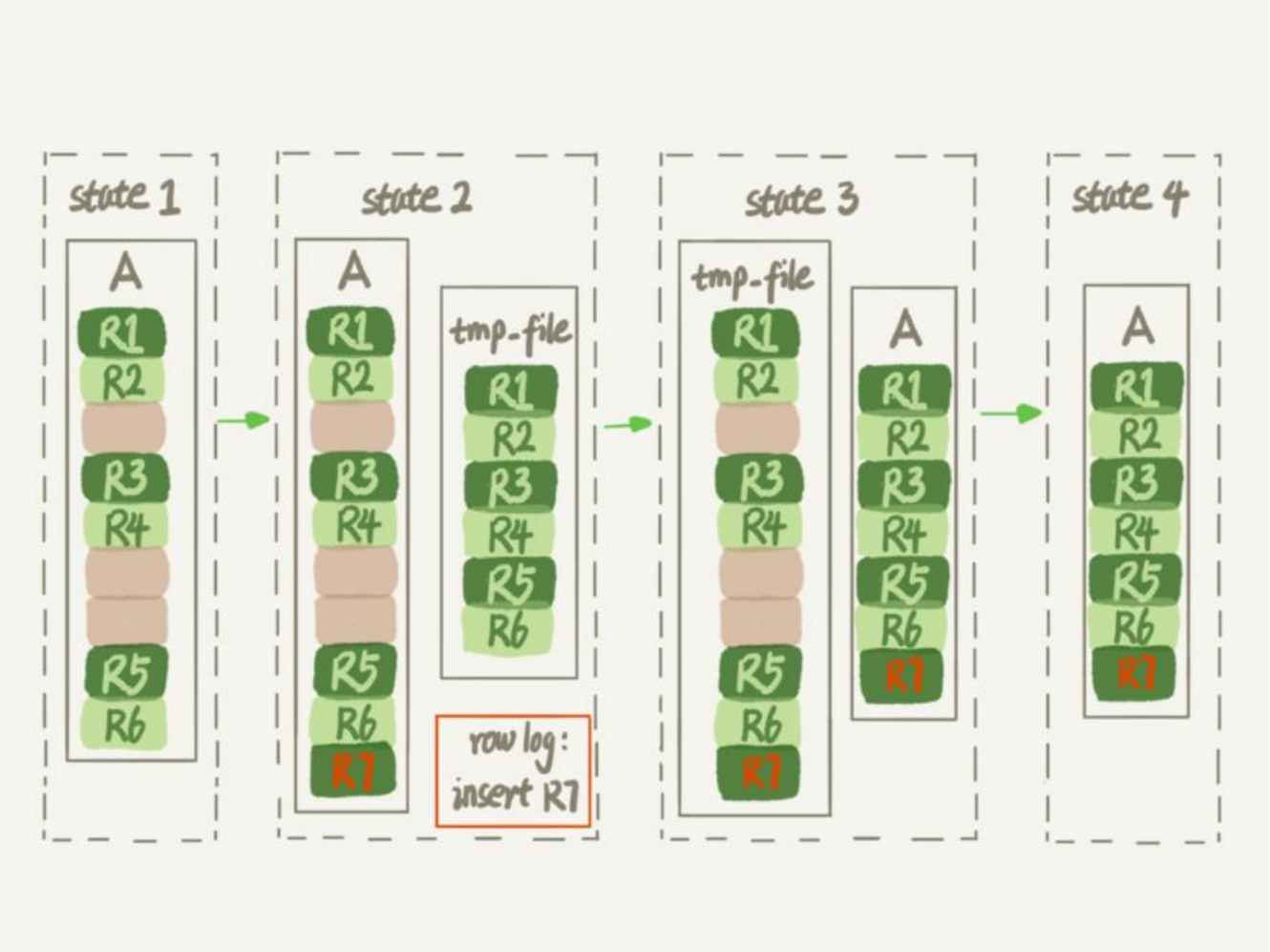
2.2 MDL锁

执行增删改查语句(DML语句)时自动会加上MDL读锁;

执行表结构更改语句(DDL)时自动会加上MDL写锁;

session C在获取MDL写锁时会被阻塞,session D在获取MDL读锁时受session C的影响也会被阻塞

online DDL



1. 建立一个临时文件, 扫描表A主键的所有数据页

2. 用数据页中表A的记录生成B+树, 存储到临时文件中

3. 生成临时文件的过程中, 对A的操作记录到日志文件中,

4. 临时文件生成后, 将日志文件中的操作应用到临时文件文件, 得到一个逻辑数据上与表A 相同的数据件

5. 用临时文件替换表A 的数据文件

alter语句在启动的时候事**先获取mdl写锁**, 在拷贝数据的时候就**退化成读锁**, 读锁不阻塞增删改数据, 是会阻塞其它线程的ddl语句(例如alter等), 由于mdl写锁的占用时间比较短就被认为是online

3. 行锁

innoDB事务中,行锁是在需要的时候才加上的,事务结束时才释放;

假设你负责实现一个电影票在线交易业务，顾客 A 要在影院 B 购买电影票。我们简化一点，这个业需要涉及到以下操作：

1. 从顾客 A 账户余额中扣除电影票价；
2. 给影院 B 的账户余额增加这张电影票价；
3. 记录一条交易日志。

锁竞争冲突的最大部分在于2步骤, 如果把2步骤放到最后一行,则最大限度可以减少事务之间的锁竞争

死锁

事务AB互相等待对方释放资源,即进入死锁状态,当出现死锁以后有两种策略

当进入死锁状态时,一般有两种策略:

1. 直接进入等待,直到超时,超时时间可以通过参数 innodb_lock_wait_timeout来设置,默认50s
2. 发起死锁检测,发现死锁后主动回滚其中一个事务, innodb_deadlock_detect设置为on表示开启

死锁检测时间复杂度为O(n^2),即1000个并发线程更新同一行,死锁检测操作量级为百万级

优化方案为:
1.拆行,一行数据拆多行,控制并发度
2.限流,控制同一时间的线程数
3.关闭死锁检测,会出现大量业务超时

问题

加入要删除10000行数据,有三种方式,哪种更好?
1.delete from T limit 10000;
2.在一个连接中循环执行20次 delete from T limit 500;
3.在20个连接中同时执行 delete from T limit 500;

方法1: 事务太长,长事务会造成redolog过长,响应延时会变大;并且长事务会引起锁竞争,事务阻塞
方法2: 相对较好,长事务切分为多个短事务,但会引起数据不一致,即在两次delete中插入新数据,会被误删

一般如果是自增主键的话,可以加上order by

方法3:会产生并发问题,循环等待形成死锁

查看单行数据慢

```
mysql> CREATE TABLE `t` (
  `id` int(11) NOT NULL,
  `c` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;

delimiter ;;
create procedure idata()
begin
  declare i int;
  set i=1;
  while(i<=100000) do
    insert into t values(i,i);
    set i=i+1;
  end while;
end;;
delimiter ;

call idata();
```

第一类:查询长时间不返回

原因一:等表锁(表锁//MDL锁)

解决方案:使用show processlist查看当前连接状态,不过一般情况下session A都处于sleep状态,无法定,这种情况下可以使用performance_schema和sys库(启动时设置performance_schema=on,会损失0%的性能)

```
select blocking_pid from sys.schema_table_lock_waits --找到pid ,kill即可
```

原因二: 等flush(全局锁)

```
--mysql中常见的flush,一般情况下很快
flush tables t with read lock;
flush tables with read lock;
```

session A	session B	session C
select sleep(1) from t;		
	flush tables t;	
		select * from t where id=1;

当循环调用sessionA时,sessionC会被长时间阻塞

原因三:等行锁

select * from t sys.innodb_lock_waits where locked_table='`test`.`t`'

```
mysql> select * from sys.innodb_lock_waits where locked_table='`test`.`t`'\G
*************************** 1. row ***************************
                wait_started: 2018-12-13 20:12:35
                    wait_age: 00:00:08
               wait_age_secs: 8
                locked_table: `test`.`t`
                locked_index: PRIMARY
                 locked_type: RECORD
              waiting_trx_id: 421668144410224
         waiting_trx_started: 2018-12-13 20:12:35
             waiting_trx_age: 00:00:08
     waiting_trx_rows_locked: 1
   waiting_trx_rows_modified: 0
                 waiting_pid: 8
               waiting_query: select * from t where id=1 lock in share mode
             waiting_lock_id: 421668144410224:23:4:2
           waiting_lock_mode: S
             blocking_trx_id: 1101302
                blocking_pid: 4
              blocking_query: NULL
            blocking_lock_id: 1101302:23:4:2
          blocking_lock_mode: X
        blocking_trx_started: 2018-12-13 20:01:57
            blocking_trx_age: 00:10:46
    blocking_trx_rows_locked: 1
  blocking_trx_rows_modified: 1
     sql_kill_blocking_query: KILL QUERY 4
sql_kill_blocking_connection: KILL 4
1 row in set, 3 warnings (0.00 sec)
```

第二类:查询慢

原因一: 扫描行数多,未使用索引

原因二: 其他线程在循环更新记录,导致undo log过长,获取到一开始的视图比较慢

原因三: 后台在刷脏页 如脏页比例过高, redolog写满,连坐刷脏页太多