**什么是gap lock**

Innodb record-level lock三种：record,gap,next-key locks

Record 锁住某一行

Gap lock 锁住某一段范围中的记录

Next-keylock 则是前两者加起来的效果

<http://dev.mysql.com/doc/refman/5.1/en/innodb-record-level-locks.html>

有资料说gap lock 是为了避免phantom(幻读)问题

<http://dev.mysql.com/doc/refman/5.1/en/innodb-next-key-locking.html>

但是gap lock扩大了锁定范围可能会造成一些不符合预期的现象

mysql> desc ts\_column\_log\_test

    -> ;

+------------+-------------+------+-----+---------------------+----------------+

| Field      |Type        | Null | Key |Default            | Extra          |

+------------+-------------+------+-----+---------------------+----------------+

|id         |int(11)     | NO   | PRI |NULL               | auto\_increment |

| col\_id     |int(11)     | NO   | MUL |NULL               |               |

| start\_time | timestamp   |NO   |     | 0000-00-00 00:00:00|               |

| end\_time   |timestamp   | NO   |     | 0000-00-0000:00:00|               |

| data\_time  | timestamp   |NO   |     | 0000-00-00 00:00:00|               |

| status     |varchar(30) | NO   |     |NULL               |               |

+------------+-------------+------+-----+---------------------+----------------+

6 rows in set (0.01 sec)

mysql> select \* from ts\_column\_log\_test;

+----+--------+---------------------+---------------------+---------------------+---------+

| id | col\_id |start\_time          |end\_time            |data\_time           |status  |

+----+--------+---------------------+---------------------+---------------------+---------+

|  1 |      2| 2011-12-13 11:51:11 | 2011-12-13 11:51:11 | 2011-12-09 00:00:00 | running |

|  2 |     20 |2011-12-13 11:51:16 | 2011-12-13 11:51:16 | 2011-12-09 00:00:00 | running |

|  3 |    120 |2011-12-13 11:51:20 | 2011-12-13 11:51:20 | 2011-12-09 00:00:00 | running |

+----+--------+---------------------+---------------------+---------------------+---------+

3 rows in set (0.00 sec)

开启两个不同的会话，分别执行一些语句观察一下结果：

session1

mysql> set autocommit=0;

mysql> delete from ts\_column\_log\_testwhere col\_id=10;

Query OK, 0 rows affected (0.00sec)        --此时[2,20)这个区间内的记录都已经被GAP LOCK锁住了，如果在其他事务中尝试插入这些值，则会等待

session2

mysql> set autocommit=0;

mysql> INSERT INTO ts\_column\_log\_test(col\_id, start\_time, end\_time, data\_time, status) VALUES (1, NULL, NULL,'20111209', 'running');  --成功

...

mysql> INSERT INTO ts\_column\_log\_test(col\_id, start\_time, end\_time, data\_time, status) VALUES (2, NULL, NULL,'20111209', 'running');  --等待

...

mysql> INSERT INTO ts\_column\_log\_test(col\_id, start\_time, end\_time, data\_time, status) VALUES (19, NULL, NULL,'20111209', 'running');  --等待

...

出现原因：session1中的delete语句中指定条件where col\_id=10，这时MYSQL会去扫描索引，但是这个时候delete语句获取的不是一个RECORD LOCK，而是一个NEXT-KEY LOCK。以当前值（10）为例，会向左扫描至col\_id=2这条记录，向右扫描至col\_id=20这条记录，锁定区间为前闭后开，即[2,20)。

下面是摘自官方手册里的一句话：

DELETE FROM ... WHERE ... sets an exclusivenext-key lock on every record the search encounters.

下面的链接里面有INNODB中各种不同的语句可能持有哪些锁的解释

http://dev.mysql.com/doc/refman/5.1/en/innodb-locks-set.html

**GAP LOCK 可能出现的问题：**

有时候我们会多个进程或线程并行的对同一张表进行操作，并且使用了事务，那么就可能会有问题，举个例子：

session1:

delete from ts\_column\_log\_test wherecol\_id=10;

INSERT INTO ts\_column\_log\_test (col\_id,start\_time, end\_time, data\_time, status) VALUES (10, NULL, NULL, '20111209','running');

session2:

delete from ts\_column\_log\_test wherecol\_id=11;

INSERT INTO ts\_column\_log\_test (col\_id,start\_time, end\_time, data\_time, status) VALUES (11, NULL, NULL, '20111209','running');

假设上面是你程序的两个进程需要做的操作，在没有并发的情况下，可能运行正常，因为每个事务在MYSQL中最终都是串行执行，中间并没有其他事务同时进行；可并发高了以后，可能在MYSQL中实际运行的语句顺序就会变成下面这个样子：

tx\_num time statement

111 2011-12-12 10:00:00 delete from ts\_column\_log\_test wherecol\_id=10;

222 2011-12-1210:00:00 delete from ts\_column\_log\_test where col\_id=11;

111 2011-12-12 10:00:00 INSERT INTO ts\_column\_log\_test (col\_id,start\_time, end\_time, data\_time, status) VALUES (10, NULL, NULL, '20111209','running');

222 2011-12-1210:00:00 INSERT INTO ts\_column\_log\_test (col\_id, start\_time, end\_time,data\_time, status) VALUES (11, NULL, NULL, '20111209', 'running');

这个时候，你可能就会得到错误提示ERROR 1213 (40001): Deadlock found when trying toget lock; try restarting transaction。

原因是前两条语句都已经获得了[2,20)这个区间内记录的S锁，然后两个事务又分别对该区间段内的col\_id=10这个位置请求X锁，这时就发生死锁，谁都请求不到X锁，因为互相都持有S锁。

**如何解决：**

1、改变程序中数据库操作的逻辑

2、取消GAP LOCK机制

Gap locking can be disabled explicitly.This occurs if you change the transaction isolation level to READ COMMITTED orenable the innodb\_locks\_unsafe\_for\_binlog system variable