<https://yq.aliyun.com/articles/108095?t=t1>

**能学到什么**

1. 隔离级别和锁的关系
2. 重点讲解在RR隔离级别下的加锁算法逻辑
3. 重点罗列了比较典型的几种加锁逻辑案例
4. 对insert的加锁逻辑进行了深度剖析
5. 实战中剖析加锁的全过程
6. InnoDB为什么要这样加锁

**隔离级别和算法**

* repeatable-read

1. 使用的是next-key locking

2. next-key lock = record lock + Gap lock

* read-committed

1. 使用的是 record lock

2. 当然特殊情况下( purge + unique key )，也会有Gap lock

我们接下来就以RR隔离级别来阐述，因为RC更加简单

* 锁的通用算法

RR隔离级别

1. 锁是在索引上实现的

2. 假设有一个key，有5条记录， 1，3，5，7，9. 如果where id<5 ， 那么锁住的区间不是（-∞，5），而是(-∞,1],(1,3],(3,5] 多个区间组合而成

3. RR隔离级别使用的是：next-key lock算法，即：锁住 记录本身+区间

4. next-key lock 降级为 record lock的情况

如果是唯一索引，且查询条件得到的结果集是1条记录（等值，而不是范围），那么会降级为记录锁

典型的案例：where primary\_key = 1 (会降级), 而不是 where primary\_key < 10 （由于返回的结果集不仅仅一条，那么不会降级）

5. 上锁，不仅仅对主键索引加锁，还需要对辅助索引加锁，这一点非常重要

**锁算法的案例剖析**

RR隔离级别

* 表结构

dba:lc\_3> show create table a;

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

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

| Table | Create Table

|

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

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

| a | CREATE TABLE `a` (

`a` int(11) NOT NULL,

`b` int(11) DEFAULT NULL,

`c` int(11) DEFAULT NULL,

`d` int(11) DEFAULT NULL,

PRIMARY KEY (`a`),

UNIQUE KEY `idx\_b` (`b`),

KEY `idx\_c` (`c`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 |

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

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

1 row in set (0.00 sec)

dba:lc\_3> select \* from a;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

| 3 | 5 | 7 | 9 |

| 5 | 7 | 9 | 11 |

| 7 | 9 | 11 | 13 |

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

4 rows in set (0.00 sec)

\* 设置RR隔离级别

set tx\_isolation = 'repeatable-read';

* 等值查询，非唯一索引的加锁逻辑

dba:lc\_3> begin;

Query OK, 0 rows affected (0.00 sec)

dba:lc\_3> select \* from a where c=9 for update;

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

| a | b | c | d |

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

| 5 | 7 | 9 | 11 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601815 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601815 lock\_mode X

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000005; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601815 lock\_mode X locks rec but not gap

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d012a; asc ' \*;;

3: len 4; hex 80000007; asc ;;

4: len 4; hex 80000009; asc ;;

5: len 4; hex 8000000b; asc ;;

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601815 lock\_mode X locks gap before rec

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 8000000b; asc ;;

1: len 4; hex 80000007; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock，((7,3),(9,5)] , ((9,5),(11,7)]，解读一下：((7,3),(9,5)] 表示：7是二级索引key，3是对应的主键

2.这样写不太好懂，所以以后就暂时忽略掉主键这样写： next-key lock = (7,9],(9,11]

对主键索引primary： 加record lock，[5]

* 等值查询，唯一键的加锁逻辑

dba:lc\_3> select \* from a where b=9 for update;

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

| a | b | c | d |

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

| 7 | 9 | 11 | 13 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601816 lock mode IX

RECORD LOCKS space id 281 page no 4 n bits 72 index idx\_b of table `lc\_3`.`a` trx id 133601816 lock\_mode X locks rec but not gap

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000007; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601816 lock\_mode X locks rec but not gap

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0137; asc ' 7;;

3: len 4; hex 80000009; asc ;;

4: len 4; hex 8000000b; asc ;;

5: len 4; hex 8000000d; asc ;;

锁的结构如下：

对二级索引idx\_b：

1. 加record lock，[9]

对主键索引primary：

1. 加record lock，[7]

* = ，非唯一索引的加锁逻辑

dba:lc\_3> select \* from a where c>=9 for update;

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

| a | b | c | d |

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

| 5 | 7 | 9 | 11 |

| 7 | 9 | 11 | 13 |

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

2 rows in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601817 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601817 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000005; asc ;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 8000000b; asc ;;

1: len 4; hex 80000007; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601817 lock\_mode X locks rec but not gap

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d012a; asc ' \*;;

3: len 4; hex 80000007; asc ;;

4: len 4; hex 80000009; asc ;;

5: len 4; hex 8000000b; asc ;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0137; asc ' 7;;

3: len 4; hex 80000009; asc ;;

4: len 4; hex 8000000b; asc ;;

5: len 4; hex 8000000d; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock， (7,9],(9,11],(11,∞]

对主键索引primary：

1. 加record lock，[5],[7]

* = ，唯一索引的加锁逻辑

dba:lc\_3> select \* from a where b>=7 for update;

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

| a | b | c | d |

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

| 5 | 7 | 9 | 11 |

| 7 | 9 | 11 | 13 |

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

2 rows in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601820 lock mode IX

RECORD LOCKS space id 281 page no 4 n bits 72 index idx\_b of table `lc\_3`.`a` trx id 133601820 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 4; hex 80000005; asc ;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000007; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601820 lock\_mode X locks rec but not gap

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d012a; asc ' \*;;

3: len 4; hex 80000007; asc ;;

4: len 4; hex 80000009; asc ;;

5: len 4; hex 8000000b; asc ;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0137; asc ' 7;;

3: len 4; hex 80000009; asc ;;

4: len 4; hex 8000000b; asc ;;

5: len 4; hex 8000000d; asc ;;

锁的结构如下：

对二级索引idx\_b：

1. 加next-key lock， (5,7],(7,9],(9,∞]

对主键索引primary：

1. 加record lock，[5],[7]

* <= , 非唯一索引的加锁逻辑

dba:lc\_3> select \* from a where c<=7 for update;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

| 3 | 5 | 7 | 9 |

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

2 rows in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601822 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601822 lock\_mode X

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 4; hex 80000001; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 4; hex 80000003; asc ;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000005; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601822 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000001; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0110; asc ' ;;

3: len 4; hex 80000003; asc ;;

4: len 4; hex 80000005; asc ;;

5: len 4; hex 80000007; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000003; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d011d; asc ' ;;

3: len 4; hex 80000005; asc ;;

4: len 4; hex 80000007; asc ;;

5: len 4; hex 80000009; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock， (-∞,5],(5,7],(7,9]

对主键索引primary：

1. 加record lock，[1],[3]

* <= , 唯一索引的加锁逻辑

dba:lc\_3> select \* from a where b<=5 for update;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

| 3 | 5 | 7 | 9 |

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

2 rows in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601823 lock mode IX

RECORD LOCKS space id 281 page no 4 n bits 72 index idx\_b of table `lc\_3`.`a` trx id 133601823 lock\_mode X

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000003; asc ;;

1: len 4; hex 80000001; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 4; hex 80000003; asc ;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 4; hex 80000005; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601823 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000001; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0110; asc ' ;;

3: len 4; hex 80000003; asc ;;

4: len 4; hex 80000005; asc ;;

5: len 4; hex 80000007; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000003; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d011d; asc ' ;;

3: len 4; hex 80000005; asc ;;

4: len 4; hex 80000007; asc ;;

5: len 4; hex 80000009; asc ;;

锁的结构如下：

对二级索引idx\_b：

1. 加next-key lock， (-∞,3],(3,5],(5,7]

对主键索引primary：

1. 加record lock，[1],[3]

* , 非唯一索引的加锁逻辑

dba:lc\_3> select \* from a where c>9 for update;

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

| a | b | c | d |

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

| 7 | 9 | 11 | 13 |

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

1 row in set (0.00 sec)

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601825 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 8000000b; asc ;;

1: len 4; hex 80000007; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601825 lock\_mode X locks rec but not gap

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0137; asc ' 7;;

3: len 4; hex 80000009; asc ;;

4: len 4; hex 8000000b; asc ;;

5: len 4; hex 8000000d; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock， (9,11],(11,∞]

对主键索引primary：

1. 加record lock，[7]

* , 唯一索引的加锁逻辑

dba:lc\_3> select \* from a where b>7 for update;

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

| a | b | c | d |

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

| 7 | 9 | 11 | 13 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601826 lock mode IX

RECORD LOCKS space id 281 page no 4 n bits 72 index idx\_b of table `lc\_3`.`a` trx id 133601826 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000007; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601826 lock\_mode X locks rec but not gap

Record lock, heap no 5 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0137; asc ' 7;;

3: len 4; hex 80000009; asc ;;

4: len 4; hex 8000000b; asc ;;

5: len 4; hex 8000000d; asc ;;

锁的结构如下：

对二级索引idx\_b：

1. 加next-key lock， (7,9],(9,∞]

对主键索引primary：

1. 加record lock，[7]

* < , 非唯一索引的加锁逻辑

dba:lc\_3> select \* from a where c<7 for update;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601827 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601827 lock\_mode X

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 4; hex 80000001; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 4; hex 80000003; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601827 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000001; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0110; asc ' ;;

3: len 4; hex 80000003; asc ;;

4: len 4; hex 80000005; asc ;;

5: len 4; hex 80000007; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock， (-∞,5],(5,7]

对主键索引primary：

1. 加record lock，[1]

* < , 唯一索引的加锁逻辑

dba:lc\_3> select \* from a where b<5 for update;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`a` trx id 133601828 lock mode IX

RECORD LOCKS space id 281 page no 4 n bits 72 index idx\_b of table `lc\_3`.`a` trx id 133601828 lock\_mode X

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000003; asc ;;

1: len 4; hex 80000001; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 4; hex 80000003; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601828 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000001; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0110; asc ' ;;

3: len 4; hex 80000003; asc ;;

4: len 4; hex 80000005; asc ;;

5: len 4; hex 80000007; asc ;;

锁的结构如下：

对二级索引idx\_c：

1. 加next-key lock， (-∞,3],(3,5]

对主键索引primary：

1. 加record lock，[1]

* 总结之前的加锁逻辑

\* 如果

1. select \* from xx where col <比较运算符> M for update

2. M->next-rec: 表示M的下一条记录

3. M->pre-rec: 表示M的前一条记录

########第一轮总结########

\* 等值查询M，非唯一索引的加锁逻辑

(M->pre-rec,M],(M,M->next-rec]

\* 等值查询M，唯一键的加锁逻辑

[M], next-lock 降级为 record locks

\* >= ，非唯一索引的加锁逻辑

(M->pre\_rec,M],(M,M->next-rec]....(∞]

\* >= ，唯一索引的加锁逻辑

(M->pre\_rec,M],(M,M->next-rec]....(∞]

\* <= , 非唯一索引的加锁逻辑

(-∞] ... (M,M->next-rec]

\* <= , 唯一索引的加锁逻辑

(-∞] ... (M,M->next-rec]

\* > , 非唯一索引的加锁逻辑

(M,M->next-rec] ... (∞]

\* > , 唯一索引的加锁逻辑

(M,M->next-rec] ... (∞]

\* < , 非唯一索引的加锁逻辑

(-∞] ... (M->rec,M]

\* < , 唯一索引的加锁逻辑

(-∞] ... (M->rec,M]

########第二轮总结合并########

\* 等值查询M，非唯一索引的加锁逻辑

(M->pre-rec,M],(M,M->next-rec]

\* 等值查询M，唯一键的加锁逻辑

[M], next-lock 降级为 record locks

这里大家还记得之前讲过的通用算法吗：

next-key lock 降级为 record lock的情况：

如果是唯一索引，且查询条件得到的结果集是1条记录（等值，而不是范围），那么会降级为记录锁

\* >= ，加锁逻辑

(M->pre\_rec,M],(M,M->next-rec]....(∞]

\* > , 加锁逻辑

(M,M->next-rec] ... (∞]

\* <= , 加锁逻辑

(-∞] ... (M,M->next-rec]

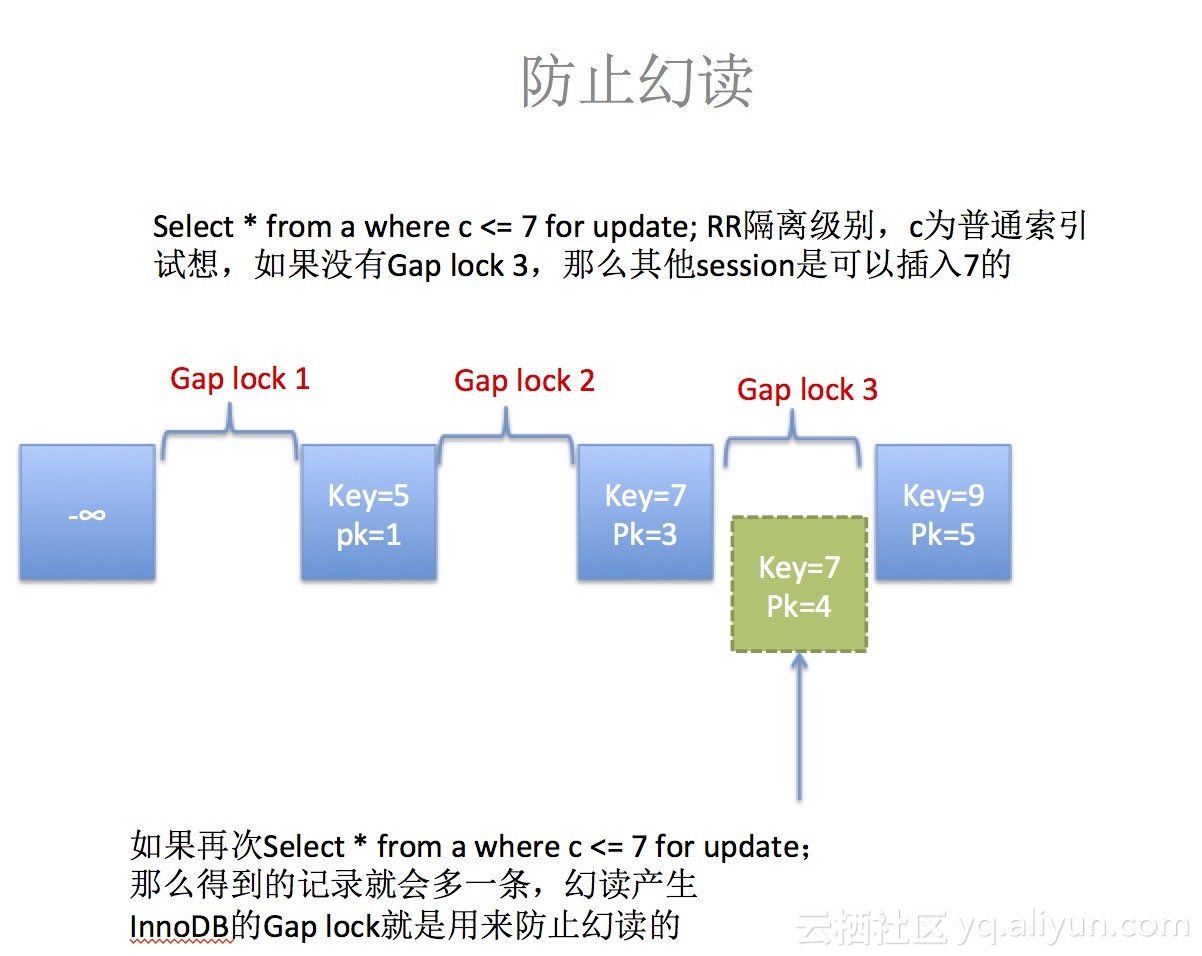
\* < , 加锁逻辑

(-∞] ... (M->rec,M]

########最后的疑问和总结########

1. 疑问： 为什么要对M->next-rec 或者 M->pre-rec ？

1. 回答： 因为为了防止幻读。



**insert 操作的加锁逻辑**

RR 隔离级别

* 表结构

dba:lc\_3> show create table tb\_non\_uk;

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

| Table | Create Table |

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

| tb\_non\_uk | CREATE TABLE `tb\_non\_uk` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`id\_2` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `idx\_id2` (`id\_2`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8 |

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

1 row in set (0.00 sec)

dba:lc\_3> show create table tb\_uk;

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

| Table | Create Table |

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

| tb\_uk | CREATE TABLE `tb\_uk` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`id\_2` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `uniq\_idx` (`id\_2`)

) ENGINE=InnoDB AUTO\_INCREMENT=36 DEFAULT CHARSET=utf8 |

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

1 row in set (0.00 sec)

dba:lc\_3> select \* from tb\_non\_uk;

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

| id | id\_2 |

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

| 1 | 100 |

| 2 | 200 |

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

2 rows in set (0.00 sec)

dba:lc\_3> select \* from tb\_uk;

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

| id | id\_2 |

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

| 1 | 10 |

| 2 | 20 |

| 33 | 30 |

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

3 rows in set (0.00 sec)

* 普通的insert,insert之前,其他事务没有对next-record加任何锁

dba:lc\_3> insert into tb\_uk select 100,200;

Query OK, 1 row affected (0.00 sec)

Records: 1 Duplicates: 0 Warnings: 0

锁的结构：

MySQL thread id 11888, OS thread handle 140000862643968, query id 24975 localhost dba cleaning up

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601936 lock mode IX

没有加任何的锁，除了在表上面加了意向锁之外，这个锁基本上只要访问到表都会加的

难道insert不会加锁吗？显然不是，那是因为加的是隐式类型的锁

* 有唯一键约束，insert之前，其他事务且对其next-record加了Gap-lock

\* session 1:

select \* from tb\_uk where id\_2 >= 30 for update;

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601951 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601951 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

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 80000021; asc !;;

RECORD LOCKS space id 301 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_uk` trx id 133601951 lock\_mode X locks rec but not gap

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000021; asc !;;

1: len 6; hex 000007f69a77; asc w;;

2: len 7; hex ad00000d010110; asc ;;

3: len 4; hex 8000001e; asc ;;

锁住： (20,30](30,∞) ， 对30有Gap锁

\* session 2:

dba:lc\_3> insert into tb\_uk select 3,25;

Query OK, 1 row affected (6.30 sec)

Records: 1 Duplicates: 0 Warnings: 0

\* session 1:

rollback;

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601952 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601952 lock\_mode X locks gap before rec insert intention

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 80000021; asc !;;

当session2 插入25的时候，这时候session2 会被卡住。 然后session 2 释放gap lock后，session 1 就持有插入意向锁 lock\_mode X locks gap before rec insert intention

* 有唯一键约束，insert之前，其他事务且对其next-record加了record lock

\* session 1:

dba:lc\_3> select \* from tb\_uk where id\_2 = 30 for update;

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

| id | id\_2 |

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

| 33 | 30 |

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

1 row in set (0.00 sec)

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601943 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

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 80000021; asc !;;

RECORD LOCKS space id 301 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000021; asc !;;

1: len 6; hex 000007f69a77; asc w;;

2: len 7; hex ad00000d010110; asc ;;

3: len 4; hex 8000001e; asc ;;

\* session 2:

dba:lc\_3> insert into tb\_uk select 3,25;

Query OK, 1 row affected (0.00 sec)

Records: 1 Duplicates: 0 Warnings: 0

锁结构：

说明有唯一键约束，insert之前，其他事务且对其next-record加了record lock，不会阻塞insert。

此时的insert，也不会产生insert intension lock

* 有唯一键约束，insert 记录之后，发现原来的表有重复值的情况,

\* session 1:

dba:lc\_3> select \* from tb\_uk where id\_2 = 30 for update;

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

| id | id\_2 |

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

| 33 | 30 |

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

1 row in set (0.00 sec)

dba:lc\_3> delete from tb\_uk where id\_2 = 20;

Query OK, 1 row affected (0.00 sec)

这时候的锁结构如下：

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601943 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

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 80000021; asc !;;

RECORD LOCKS space id 301 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 32

0: len 4; hex 80000002; asc ;;

1: len 6; hex 000007f69a97; asc ;;

2: len 7; hex 460000403f090b; asc F @? ;;

3: len 4; hex 80000014; asc ;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000021; asc !;;

1: len 6; hex 000007f69a77; asc w;;

2: len 7; hex ad00000d010110; asc ;;

3: len 4; hex 8000001e; asc ;;

对二级索引uniq\_idx ：

1. 加record lock ， [20]，[30]

对主键索引：

1. 加record lock，[2],[33]

\* session 2:

dba:lc\_3> insert into tb\_uk select 3,20;

...............waiting.................

这时候，我们再来看看锁结构：

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601949 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601949 lock mode S waiting

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 ;;

---TRANSACTION 133601943, ACTIVE 490 sec

3 lock struct(s), heap size 1136, 4 row lock(s), undo log entries 1

MySQL thread id 11889, OS thread handle 140000878618368, query id 25018 localhost dba cleaning up

TABLE LOCK table `lc\_3`.`tb\_uk` trx id 133601943 lock mode IX

RECORD LOCKS space id 301 page no 4 n bits 72 index uniq\_idx of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

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 80000021; asc !;;

RECORD LOCKS space id 301 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_uk` trx id 133601943 lock\_mode X locks rec but not gap

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 32

0: len 4; hex 80000002; asc ;;

1: len 6; hex 000007f69a97; asc ;;

2: len 7; hex 460000403f090b; asc F @? ;;

3: len 4; hex 80000014; asc ;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000021; asc !;;

1: len 6; hex 000007f69a77; asc w;;

2: len 7; hex ad00000d010110; asc ;;

3: len 4; hex 8000001e; asc ;;

info bits 32 表示这条记录已经标记为删除状态

这里面的session 2 ： insert into tb\_uk select 3,20; 被阻塞了

因为，这条insert 语句需要对 uniq\_idx中的20加lock mode S ， 但是发现session 1 已经对其加了lock\_mode X locks rec but not gap，而这条记录被标记为删除状态

所以发生锁等待，因为S lock 和 X lock 冲突

* 没有唯一键约束,insert之前，其他事务对其next-record加了Gap-lock

\* session 1:

dba:lc\_3> select \* from tb\_non\_uk where id\_2>=100 for update;

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

| id | id\_2 |

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

| 1 | 100 |

| 2 | 200 |

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

2 rows in set (0.00 sec)

锁结构：

TABLE LOCK table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock mode IX

RECORD LOCKS space id 302 page no 4 n bits 72 index idx\_id2 of table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 800000c8; asc ;;

1: len 4; hex 80000002; asc ;;

RECORD LOCKS space id 302 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock\_mode X locks rec but not gap

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000002; asc ;;

1: len 6; hex 000007f69a6b; asc k;;

2: len 7; hex a500000d360110; asc 6 ;;

3: len 4; hex 800000c8; asc ;;

对idx\_id2二级索引： (100,200],(200,∞]

对主键索引： [2]

\* session 2:

dba:lc\_3> insert into tb\_non\_uk select 3,150;

......waiting.....

---TRANSACTION 133601940, ACTIVE 3 sec inserting

mysql tables in use 1, locked 1

LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s), undo log entries 1

MySQL thread id 11888, OS thread handle 140000862643968, query id 24996 localhost dba executing

insert into tb\_non\_uk select 3,150

------- TRX HAS BEEN WAITING 3 SEC FOR THIS LOCK TO BE GRANTED:

RECORD LOCKS space id 302 page no 4 n bits 72 index idx\_id2 of table `lc\_3`.`tb\_non\_uk` trx id 133601940 lock\_mode X locks gap before rec insert intention waiting

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 800000c8; asc ;;

1: len 4; hex 80000002; asc ;;

------------------

TABLE LOCK table `lc\_3`.`tb\_non\_uk` trx id 133601940 lock mode IX

RECORD LOCKS space id 302 page no 4 n bits 72 index idx\_id2 of table `lc\_3`.`tb\_non\_uk` trx id 133601940 lock\_mode X locks gap before rec insert intention waiting

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 800000c8; asc ;;

1: len 4; hex 80000002; asc ;;

---TRANSACTION 133601939, ACTIVE 311 sec

3 lock struct(s), heap size 1136, 3 row lock(s)

MySQL thread id 11889, OS thread handle 140000878618368, query id 24994 localhost dba cleaning up

TABLE LOCK table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock mode IX

RECORD LOCKS space id 302 page no 4 n bits 72 index idx\_id2 of table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 800000c8; asc ;;

1: len 4; hex 80000002; asc ;;

RECORD LOCKS space id 302 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`tb\_non\_uk` trx id 133601939 lock\_mode X locks rec but not gap

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 4; hex 80000002; asc ;;

1: len 6; hex 000007f69a6b; asc k;;

2: len 7; hex a500000d360110; asc 6 ;;

3: len 4; hex 800000c8; asc ;;

锁结构：

多了一个插入意向锁 lock\_mode X locks gap before rec insert intention

* 总结Insert 操作的加锁流程

\* insert 的流程(没有唯一索引的情况)： insert N

1. 找到大于N的第一条记录M

2. 如果M上面没有gap ， next-key locking的话，可以插入 ， 否则等待 (对其next-rec加insert intension lock，由于有gap锁，所以等待)

\* insert 的流程(有唯一索引的情况)： insert N

1. 找到大于N的第一条记录M，以及前一条记录P

2. 如果M上面没有gap ， next-key locking的话，进入第三步骤 ， 否则等待(对其next-rec加insert intension lock，由于有gap锁，所以等待)

3. 检查p：

判断p是否等于n：

如果不等: 则完成插入（结束）

如果相等：

再判断P 是否有锁，

如果没有锁:

报1062错误（duplicate key） --说明该记录已经存在，报重复值错误

加S-lock --说明该记录被标记为删除, 事务已经提交，还没来得及purge

如果有锁: 则加S-lock --说明该记录被标记为删除，事务还未提交.

\* insert intension lock 有什么用呢？锁的兼容矩阵是啥？

1. insert intension lock 是一种特殊的Gap lock，记住非常特殊哦

2. insert intension lock 和 insert intension lock 是兼容的，其次都是不兼容的

3. Gap lock 是为了防止insert， insert intension lock 是为了insert并发更快，两者是有区别的

4. 什么情况下会出发insert intension lock ？

当insert的记录M的 next-record 加了Gap lock才会发生，record lock并不会触发

**实战案例**

RR 隔离级别  
最后来一个比较复杂的案例作为结束  
通过这几个案例，可以复习下之前讲过的理论，锁不仅对主键加，还要考虑二级索引哦

* 环境

set tx\_isolation = 'repeatable-read';

CREATE TABLE `a` (

`a` int(11) NOT NULL,

`b` int(11) DEFAULT NULL,

`c` int(11) DEFAULT NULL,

`d` int(11) DEFAULT NULL,

PRIMARY KEY (`a`),

UNIQUE KEY `idx\_b` (`b`),

KEY `idx\_c` (`c`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

dba:lc\_3> select \* from a;

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

| a | b | c | d |

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

| 1 | 3 | 5 | 7 |

| 3 | 5 | 7 | 9 |

| 5 | 7 | 9 | 11 |

| 7 | 9 | 11 | 13 |

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

4 rows in set (0.00 sec)

* 加锁语句

select \* from a where c<9 for update;

锁结构：

TABLE LOCK table `lc\_3`.`a` trx id 133601957 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601957 lock\_mode X

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000005; asc ;;

1: len 4; hex 80000001; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000007; asc ;;

1: len 4; hex 80000003; asc ;;

Record lock, heap no 4 PHYSICAL RECORD: n\_fields 2; compact format; info bits 0

0: len 4; hex 80000009; asc ;;

1: len 4; hex 80000005; asc ;;

RECORD LOCKS space id 281 page no 3 n bits 72 index PRIMARY of table `lc\_3`.`a` trx id 133601957 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000001; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d0110; asc ' ;;

3: len 4; hex 80000003; asc ;;

4: len 4; hex 80000005; asc ;;

5: len 4; hex 80000007; asc ;;

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 6; compact format; info bits 0

0: len 4; hex 80000003; asc ;;

1: len 6; hex 000007f66444; asc dD;;

2: len 7; hex fc0000271d011d; asc ' ;;

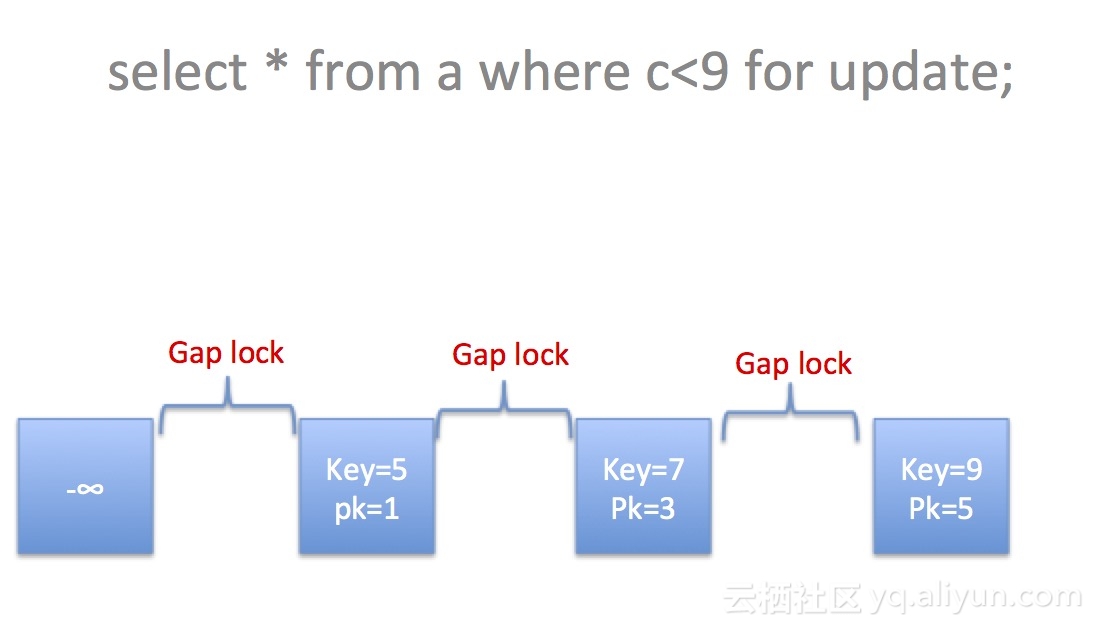
3: len 4; hex 80000005; asc ;;

4: len 4; hex 80000007; asc ;;

5: len 4; hex 80000009; asc ;;

二级索引idx\_c 加锁 next-key lock： (-∞,5],(5,7],(7,9]

primary key 加锁 record lock： [1]和[3]



* 案例一 insert into a select 4,40,9,90

大家觉得能够插入成功吗？

dba:lc\_3> insert into a select 4,40,9,90;

^C^C -- query aborted

ERROR 1317 (70100): Query execution was interrupted

...................waiting.................

显然是被锁住了

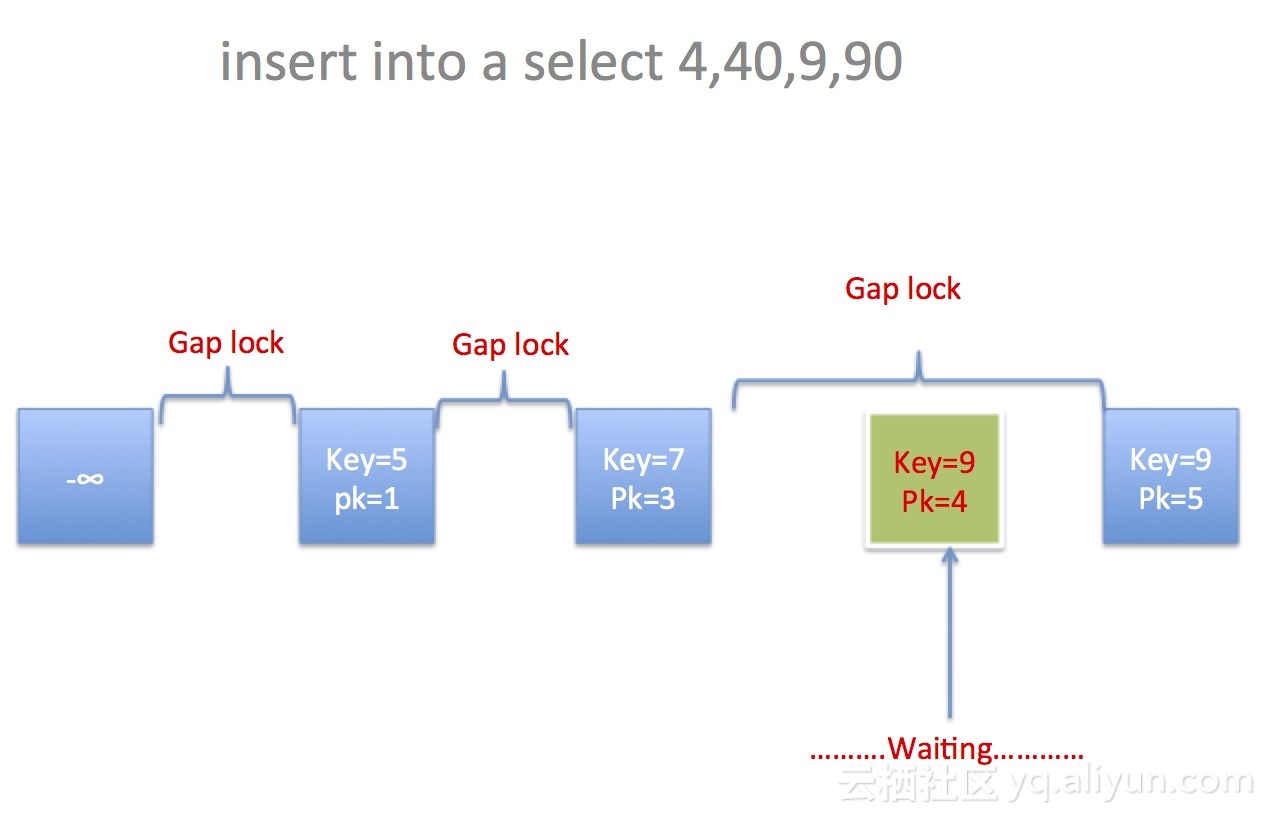
TABLE LOCK table `lc\_3`.`a` trx id 133601961 lock mode IX

RECORD LOCKS space id 281 page no 5 n bits 72 index idx\_c of table `lc\_3`.`a` trx id 133601961 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 80000009; asc ;;

1: len 4; hex 80000005; asc ;;



* 案例二 insert into a select 6,40,9,90;

大家觉得能够插入成功吗？

dba:lc\_3> insert into a select 6,40,9,90;

Query OK, 1 row affected (0.00 sec)

Records: 1 Duplicates: 0 Warnings: 0

显然是插入成功了

