链滴

Mysql 各种锁

作者：xiaokedamowang
原文链接：https://ld246.com/article/1628941816064
来源网站：链滴

Mysql 各种锁

真是呕心沥血查资料, 做试验, 做分析, 可怜.jpg

一. 共享锁和排他锁(读锁和写锁)

共享锁又叫读锁

排他锁又叫写锁

锁和锁的关系如下表格:

	共享锁	排他锁
共享锁	√	×
排他锁	×	×

以下都叫读锁和写锁了, 可以少打几个字

二. 表锁

就是加在表上的锁, 上锁方式如下

1. 手动上表锁

```
# 多个用逗号分隔
# read就是表级读锁
# write就是表级写锁
LOCK tables orders read LOCAL,order_detail write;
LOCK tables orders read,order_detail write;
# LOCAL 可以加也可以不加
# 加了LOCAL 其他不能修改, 但是可以在表尾插入新的数据, 也就是ID一定要在后面加新记录, 不能加中间
```

2. **当表没有主键或唯一索引, 或者表有索引 但是sql索引没有生效的时候, 也会上表锁**

当A连接加了 **读** 表锁之后, A连接自己无法读取和修改其他表,并且无法当前表做修改, **其他连接**可以读取该表的数据, 但是不能修改, 如果修改, 会阻塞等待, 直到锁释放

当A连接加了 **写** 表锁之后, A连接自己无法读取和修改别的表, **其他连接**和写都需要阻塞等待, 直到锁释放

三. 行锁

就是加在记录上的锁, 上锁方式如下

1. 手动加行锁

```
#加写锁 只给id为1的记录加
select * from tables where id = 1 for update
```

```
#加写锁 给全部记录加
select * from tables for update
#加读锁 给id为1的记录加
select * from tables  where id = 1 lock in share mode
#加读锁 给全部记录加
select * from tables lock in share mode
```

2. 增删改自动加行的写锁

修改和删除好理解, 增加记录还会有别的锁

行锁的互斥关系和[一. 共享锁和排他锁(读锁和写锁)](# 一. 共享锁和排他锁(读锁和写锁))是一样的

如果一条记录加了读锁, 那么别的连接只能读, 不能写, 写的话就会阻塞

如果一条记录加了写锁, 那么别的连接不能读也不能写, 读写都会阻塞

四. 临键锁(next_key) 和 间隙锁

临键锁(next_key) 和 间隙锁 是为了在RR隔离级别下解决幻读问题, 直接把区间锁住, 记录就插入不了, 就不会产生幻读

间隙锁: 顾名思义, 锁住一段间隙

临键锁: 就是间隙锁 + 行锁, 多锁了1个数据

innodb默认使用临键锁(next_key)但是在主键索引和唯一索引, 遇到下的情况会退化锁

精确匹配	**行锁**
精确匹配值不存在	**间隙锁**
范围匹配	**行锁+间隙锁**

```
# 以下操作都在RR隔离级别上  也就是可重复读  间隙锁和临键锁 是为了解决幻读
CREATE TABLE `test_lock` (
  `id` int NOT NULL AUTO_INCREMENT,
  `age` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `age` (`age`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_b
n;

INSERT INTO `3pp_menus`.`test_lock`(`id`, `age`) VALUES (5, 5);
INSERT INTO `3pp_menus`.`test_lock`(`id`, `age`) VALUES (10, 10);
INSERT INTO `3pp_menus`.`test_lock`(`id`, `age`) VALUES (15, 15);
INSERT INTO `3pp_menus`.`test_lock`(`id`, `age`) VALUES (20, 20);
INSERT INTO `3pp_menus`.`test_lock`(`id`, `age`) VALUES (25, 25);
```

id(主键或唯一索引)	**age(普通索引)**
5	5
10	10

15 15
20 20
25 25

1. 对主键或唯一索引操作

    1. 等值查询查到记录

```
#事务A
select * from test_lock where id = 5 for update
#事务B
insert into test_lock values (4,4) #成功
insert into test_lock values (6,6) #成功
update test_lock set age = 123 where id = 5 #阻塞
```

加行锁, 只在id为5的记录上加

    2. 等值查询未查到记录

```
#事务A
select * from test_lock where id = 12 for update
#事务B
insert into test_lock values (4,4) #成功
insert into test_lock values (6,6) #成功
insert into test_lock values (11,11) #阻塞
insert into test_lock values (16,16) #成功
update test_lock set age = age + 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #成功
```

加间隙锁, (10,15) 左开又开

    3. 范围查询1

```
#事务A
select * from test_lock where id > 10 for update
#事务B
insert into test_lock values (6,6) #成功
insert into test_lock values (11,11) #阻塞
insert into test_lock values (16,16) #阻塞
update test_lock set age = age + 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #阻塞
update test_lock set age = age + 1 where id = 20 #阻塞
```

加间隙锁, (10,无穷大)

    4. 范围查询2

```
#事务A
select * from test_lock where id > 17  for update
#事务B
insert into test_lock values (14,14) #成功
insert into test_lock values (16,16) #阻塞
insert into test_lock values (18,18) #阻塞
update test_lock set age = age + 1 where id = 15 #成功
update test_lock set age = age + 1 where id = 20 #阻塞
```

加间隙锁, (15,无限大)

5. 范围查询3

```
#事务A
select * from test_lock where id > 12 and id < 17 for update
#事务B
insert into test_lock values (8,8) #成功
insert into test_lock values (11,11) #阻塞
insert into test_lock values (14,14) #阻塞
insert into test_lock values (16,16) #阻塞
insert into test_lock values (18,18) #阻塞
insert into test_lock values (21,21) #成功
update test_lock set age = age + 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #阻塞
update test_lock set age = age + 1 where id = 20 #成功
```

间隙锁 + 行锁 + 间隙锁, (10,15) [15] (15,20)

6. 范围查询4

```
#事务A
select * from test_lock where id < 17 for update
#事务B
insert into test_lock values (21,21) #成功
insert into test_lock values (18,18) #阻塞
insert into test_lock values (4,4) #阻塞
update test_lock set age = age + 1 where id = 20 #成功
update test_lock set age = age + 1 where id = 15 #阻塞
```

间隙锁 (无限小,20)

2. 对普通索引操作

先需要了解1下索引树是什么样子的, 更详细可以查看这篇博客mysql 索引

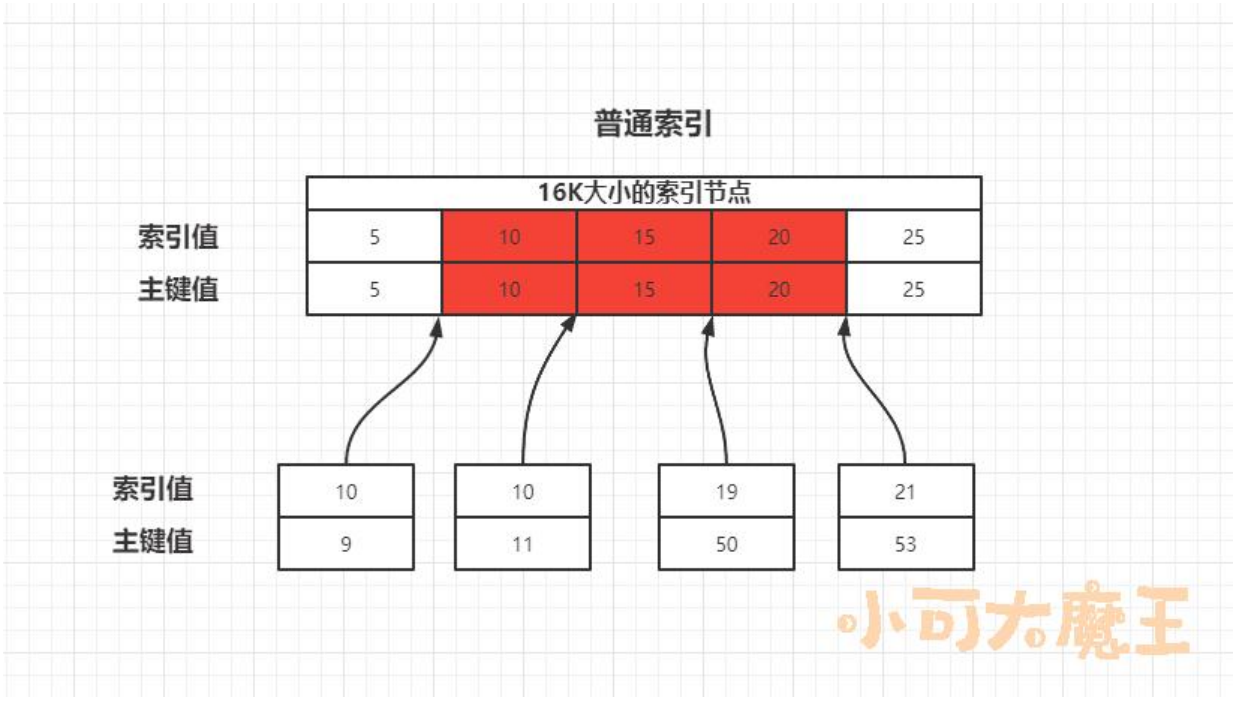
下面是我们的测试数据图, 由于数据量太少, 所以只有一个节点

1. 等值查询查到记录

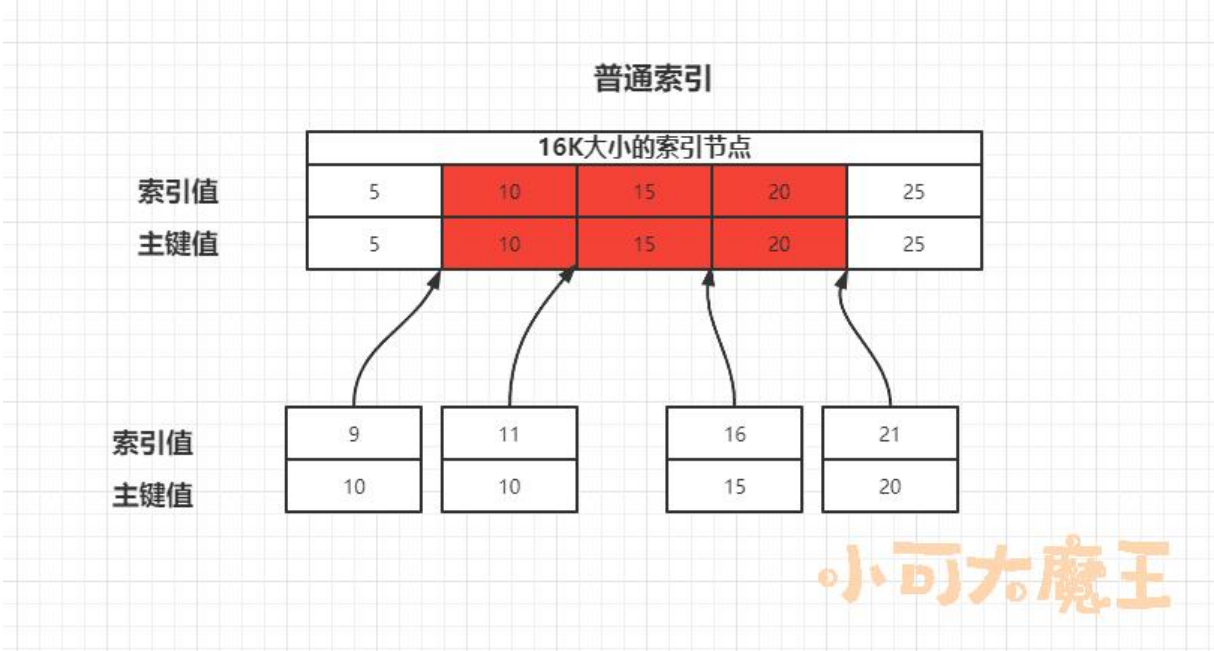
```
#事务A
select * from test_lock where age = 15 for update
#事务B
insert into test_lock (id,age)values (9,10) #成功
insert into test_lock (id,age)values (11,10) #阻塞
insert into test_lock (id,age)values (50,14) #阻塞
insert into test_lock (id,age)values (51,16) #阻塞
insert into test_lock (id,age)values (52,19) #阻塞
insert into test_lock (id,age)values (53,21) #成功
update test_lock set age = age + 1 where id = 10 #阻塞
update test_lock set age = age - 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #阻塞
update test_lock set age = age + 1 where id = 20 #成功
```

临键锁 + 间隙锁 (10,15] + (15,20)

插入的情况如下



修改的情况如下



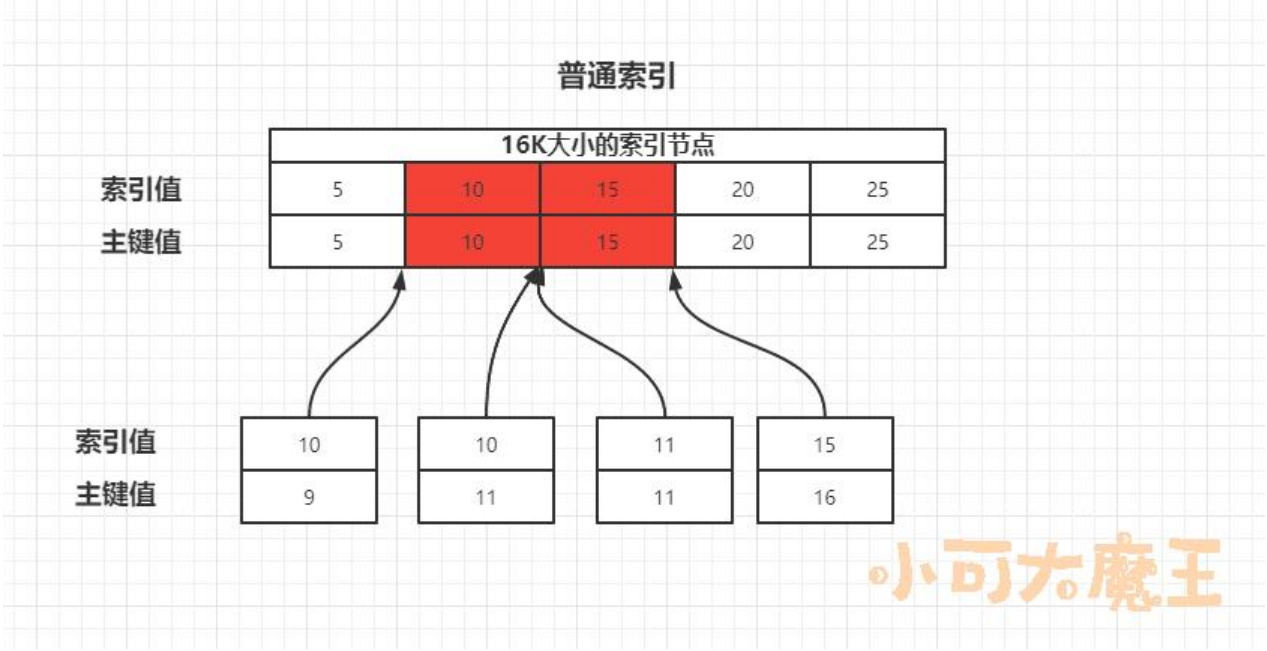
修改不太好画, 简单来说就是可以把值改出去, 但是不能改进来, 哪怕原本就在这个范围里

2. 等值查询未查到记录

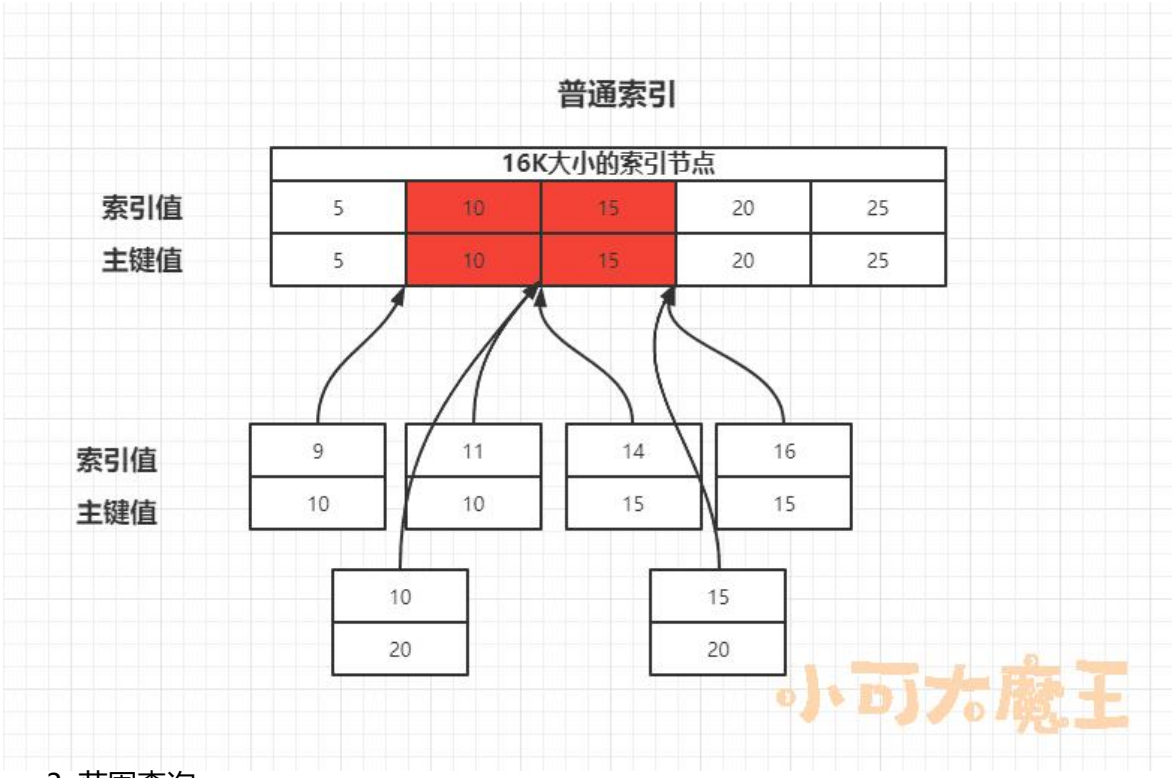
```
#事务A
select * from test_lock where age = 12 for update
#事务B
insert into test_lock (id,age)values (9,10) #成功
insert into test_lock (id,age)values (11,10) #阻塞
insert into test_lock (id,age)values (11,11) #阻塞
insert into test_lock (id,age)values (16,15) #成功
insert into test_lock (id,age)values (21,10) #阻塞
insert into test_lock (id,age)values (30,14) #阻塞
insert into test_lock (id,age)values (32,16) #成功
update test_lock set age = age + 1 where id = 10 #阻塞
update test_lock set age = age - 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #成功
update test_lock set age = age - 1 where id = 15 #阻塞
update test_lock set age = 10 where id = 20 #阻塞
update test_lock set age = 10 where id = 25 #阻塞
update test_lock set age = 15 where id = 20 #成功
update test_lock set age = 10 where id = 5 #成功
```

间隙锁 (10,15)

插入情况



修改情况



3. 范围查询

```
#事务A
select * from test_lock where age > 12 and age < 17 for update
#事务B
insert into test_lock (id,age)values (9,10) #成功
insert into test_lock (id,age)values (11,10) #阻塞
insert into test_lock (id,age)values (50,14) #阻塞
insert into test_lock (id,age)values (51,16) #阻塞
insert into test_lock (id,age)values (52,19) #阻塞
insert into test_lock (id,age)values (53,21) #成功
```

update test_lock set age = age + 1 where id = 10 #阻塞
update test_lock set age = age - 1 where id = 10 #成功
update test_lock set age = age + 1 where id = 15 #阻塞
update test_lock set age = age + 1 where id = 20 #阻塞 只有这里和案例一不一样
update test_lock set age = 20 where id = 25 #成功
update test_lock set age = 19 where id = 25 #阻塞

临键锁 + 临键锁 (10,15] (15,20]

这里只有**update test_lock set age = age + 1 where id = 20** 和案一不一样, 只是多锁了1条记录, 所以后面的间隙锁变成了临键锁

间隙锁 和 间隙锁不是互斥的, 假如2个连接都拿到了同样的间隙锁, 然后去修改新增, 那么会产生死锁问题

update, delete, select for update 可以获得间隙锁

五. 意向锁

意向锁是锁在表上的, 有意向排他锁, 和意向共享锁

主要的作用就是加速锁的判断

比如: A连接 对某条记录加行锁**(读锁, 写锁都行)**, B连接这时候如果想加表级写锁, 那么他得一条条录遍历, 才知道有没有人加过锁

那么A给某条记录加锁之前, 给表加意向锁, 代表有人正在执行**读或写**的操作, 那么B连接不用遍历也知道 有人在操作数据, 他的写锁得等前面人操作完成后才能加, 直接阻塞等待就行

情况就2种

1. A要写, 给表加意向写锁, B要加表锁, 不管是读写都得阻塞
2. A要读, 给表加意向读锁, B要加表级写锁, 阻塞等待

六.自增锁

自增锁是MySQL一种特殊的锁，如果表中存在自增字段，MySQL便会自动维护一个自增锁。

就是ID自增的时候, 要确保ID不重复的锁

1. 插入空值的时候

1、申请AUTO_INC锁
2、得到当前的AUTO_INCREMENT值n，并加1
3、执行插入操作，并将n写入新增的对应字段中。
4、释放AUTO_INC锁。

2. 插入已经有值的自增

1、插入第一条数据
2、如果失败流程结束
3、如果成功，申请AUTO_INC锁
4、调用set_max函数，修改AUTO_INCREMENT

5、语句结束，释放AUTO_INC锁

七. 乐观锁

乐观锁需要自己实现

1. 可以多加1个version字段, 每次修改的时候where条件带上version

```
select * from user where id = 10
# 查出记录, 比如: id = 10, name = 张三, age = 20, version = 3
# 然后修改张三的年龄为21
update user set age = 21, version = version + 1 where id = 10 and version = 3
# 如果有别的线程改过,并且也遵循乐观锁的规则, 每次修改数据都version + 1, 那么上面这条sql就会行失败, 因为表中的version 已经被别人 + 1了,变成了4, 找不到id = 10, version = 3的记录
```

2. 也可以直接where条件带上旧值, 不过这适合修改一个字段

```
select * from user where id = 10
# 查出记录, 比如: id = 10, name = 张三, age = 20
# 然后修改张三的年龄为21
update user set age = 21 where id = 10 and age = 20
```