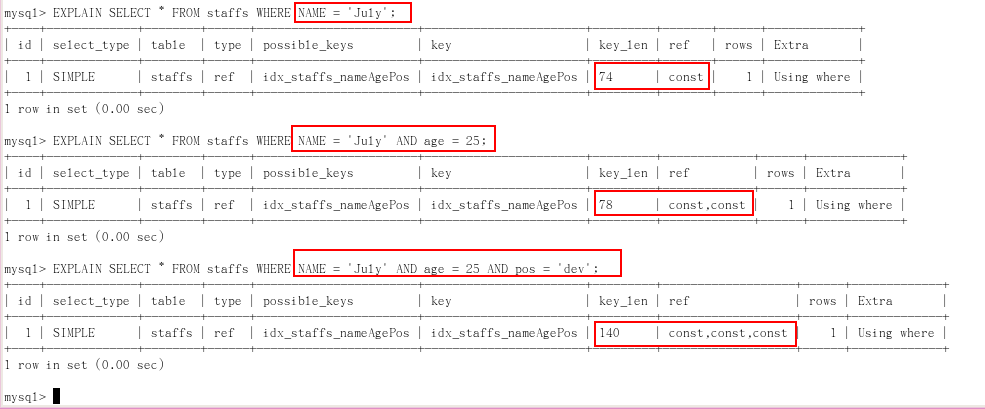
# SQL优化

## 优化实战

### 策略1.尽量全值匹配



CREATE TABLE `staffs`(

id int primary key auto\_increment,

name varchar(24) not null default "" comment'姓名',

age int not null default 0 comment '年龄',

pos varchar(20) not null default "" comment'职位',

add\_time timestamp not null default current\_timestamp comment '入职时间'

)charset utf8 comment '员工记录表';

insert into staffs(name,age,pos,add\_time) values('z3',22,'manage',now());

insert into staffs(name,age,pos,add\_time) values('july',23,'dev',now());

insert into staffs(name,age,pos,add\_time) values('2000',23,'dev',now());

alter table staffs add index idx\_staffs\_nameAgePos(name,age,pos);

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July';

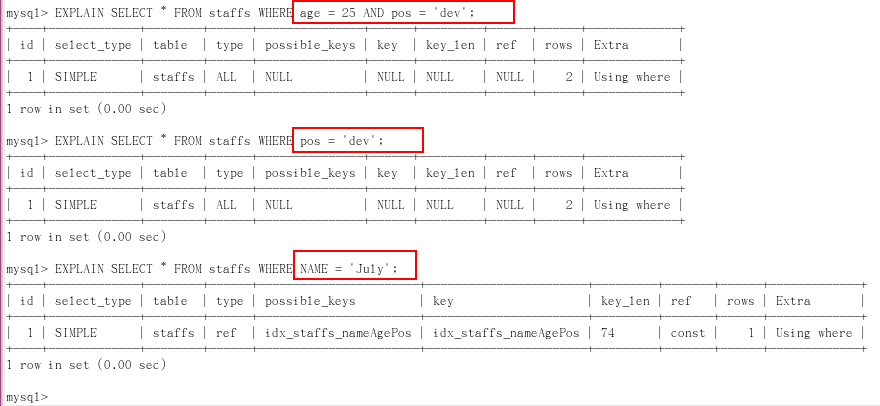
EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' AND age = 25;

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' AND age = 25 AND pos = 'dev'

当建立了索引列后，能在wherel条件中使用索引的尽量所用。

### 策略2.最佳左前缀法则

如果索引了多列，要遵守最左前缀法则。指的是查询从索引的最左前列开始并且不跳过索引中的列。



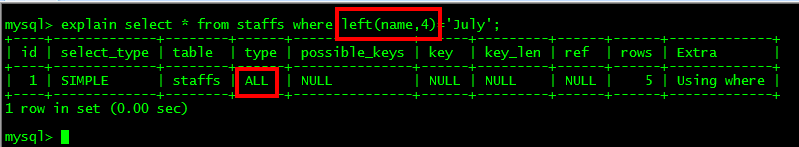
EXPLAIN SELECT \* FROM staffs WHERE age = 25 AND pos = 'dev'

EXPLAIN SELECT \* FROM staffs WHERE pos = 'dev'

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July'

### 策略3.不在索引列上做任何操作

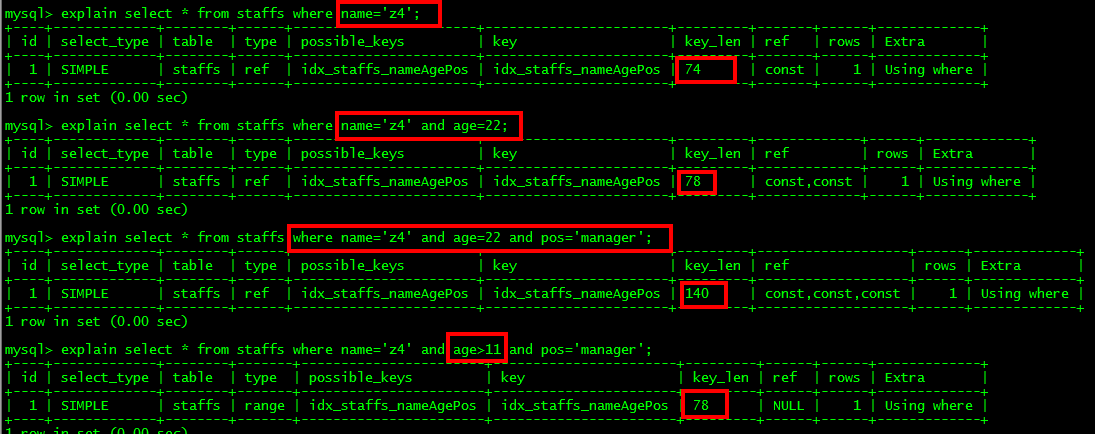
不在索引列上做任何操作（计算、函数、(自动or手动)类型转换），会导致索引失效而转向全表扫描



EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July';

EXPLAIN SELECT \* FROM staffs WHERE left(NAME,4) = 'July';

### 策略4.范围条件放最后



EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' ;

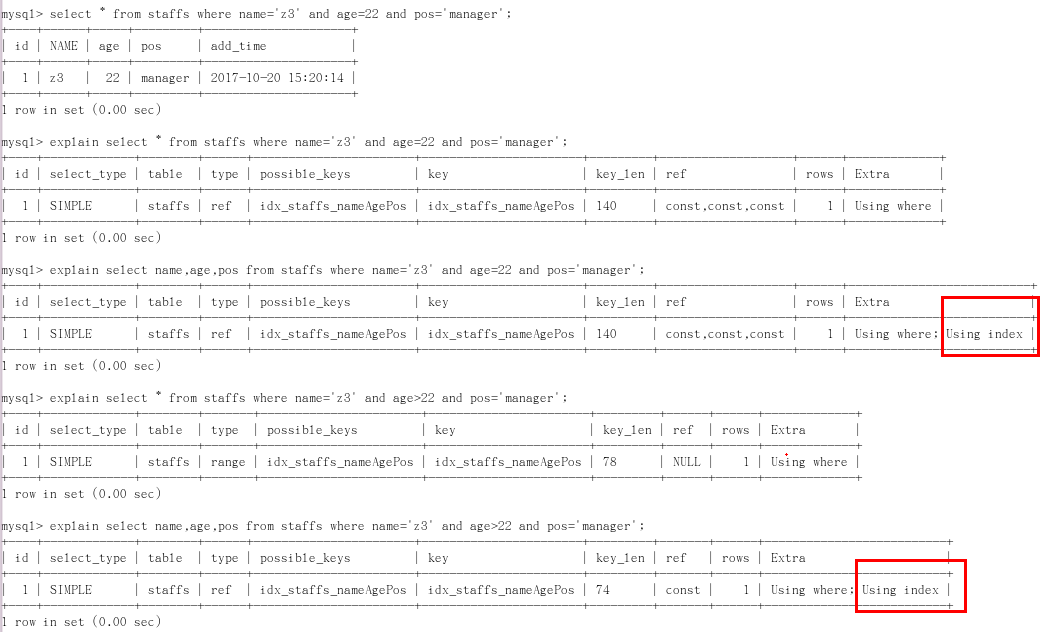
EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' and age =22;

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' and age =22 and pos='manager'

中间有范围查询会导致后面的索引列全部失效

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' and age >22 and pos='manager'

### 策略5.覆盖索引尽量用



尽量使用覆盖索引(只访问索引的查询(索引列和查询列一致))，减少select \*

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' and age =22 and pos='manager'

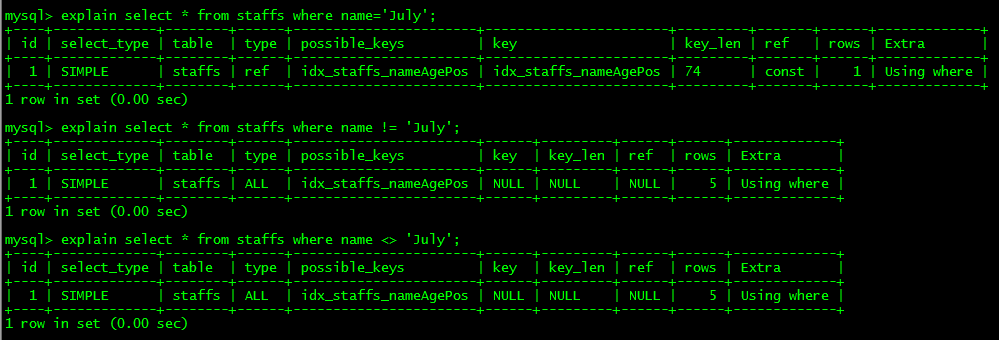
EXPLAIN SELECT name,age,pos FROM staffs WHERE NAME = 'July' and age =22 and pos='manager'

EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July' and age >22 and pos='manager'

EXPLAIN SELECT name,age,pos FROM staffs WHERE NAME = 'July' and age >22 and pos='manager'

### 策略6.不等于要甚用

mysql 在使用不等于(!= 或者<>)的时候无法使用索引会导致全表扫描



EXPLAIN SELECT \* FROM staffs WHERE NAME = 'July';

EXPLAIN SELECT \* FROM staffs WHERE NAME != 'July';

EXPLAIN SELECT \* FROM staffs WHERE NAME <> 'July';

**如果定要需要使用不等于,请用覆盖索引**

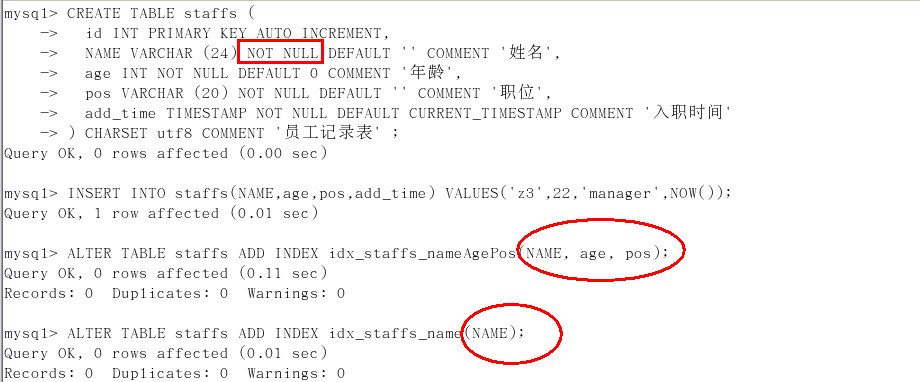
**EXPLAIN SELECT name,age,pos FROM staffs WHERE NAME != 'July';**

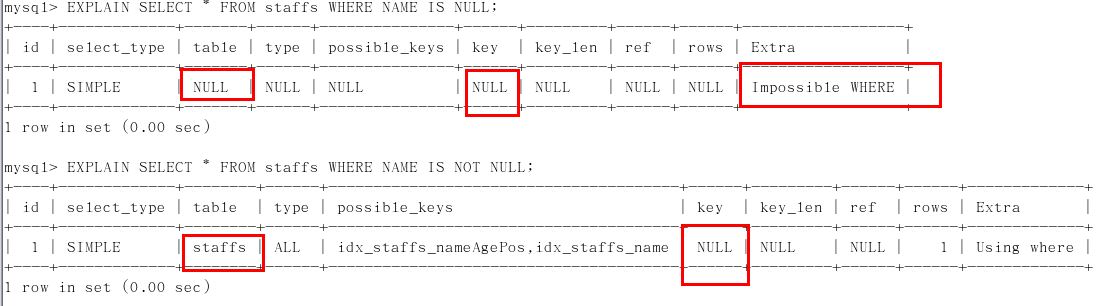
**EXPLAIN SELECT name,age,pos FROM staffs WHERE NAME <> 'July';**

### 策略7.Null/Not 有影响

**注意null/not null对索引的可能影响**

#### 自定定义为NOT NULL





EXPLAIN select \* from staffs where name is null

EXPLAIN select \* from staffs where name is not null

在字段为not null的情况下，使用is null 或 is not null 会导致索引失效

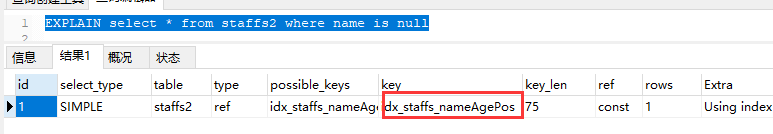
解决方式：覆盖索引

EXPLAIN select name,age,pos from staffs where name is not null

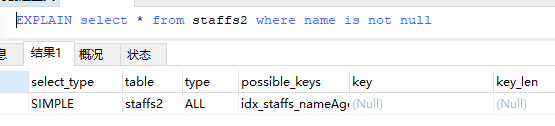
#### 自定义为NULL或者不定义



EXPLAIN select \* from staffs2 where name is null



EXPLAIN select \* from staffs2 where name is not null



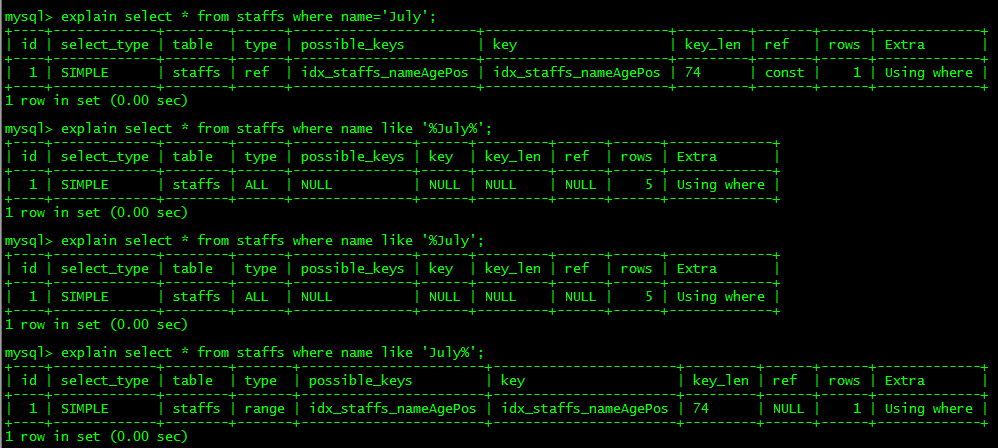
Is not null 的情况会导致索引失效

解决方式：覆盖索引

EXPLAIN select name,age,pos from staffs where name is not null

### 策略8.Like查询要当心

like以通配符开头('%abc...')mysql索引失效会变成全表扫描的操作



EXPLAIN select \* from staffs where name ='july'

EXPLAIN select \* from staffs where name like '%july%'

EXPLAIN select \* from staffs where name like '%july'

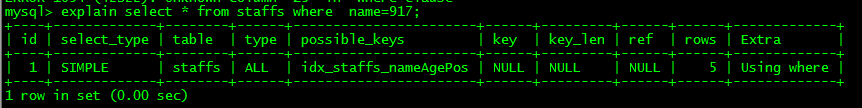
EXPLAIN select \* from staffs where name like 'july%'

解决方式：覆盖索引

EXPLAIN select name,age,pos from staffs where name like '%july%'

### 策略9.字符类型加引号

字符串不加单引号索引失效



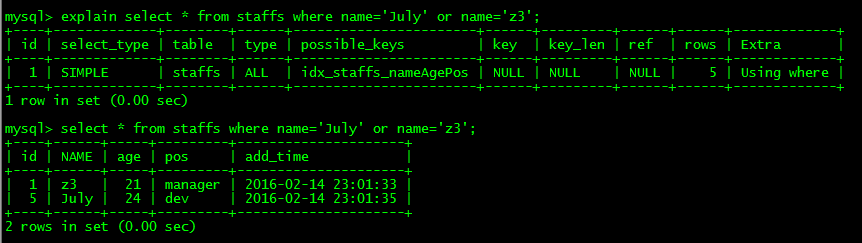
EXPLAIN select \* from staffs where name = 917

~~解决方式：覆盖索引~~

~~EXPLAIN select name,age,pos from staffs where name = 917~~

解决方式：请加引号

### 策略10.OR改UNION效率高



EXPLAIN

select \* from staffs where name='July' or name = 'z3'

EXPLAIN

select \* from staffs where name='July'

UNION

select \* from staffs where name = 'z3'

解决方式：覆盖索引

EXPLAIN

select name,age from staffs where name='July' or name = 'z3'

### 测试题



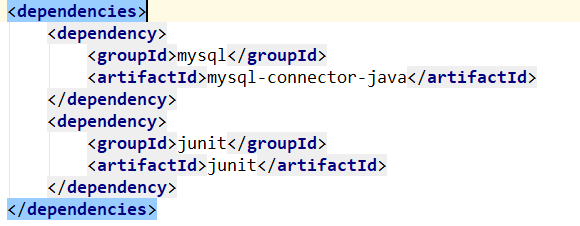
答案：



记忆总结：

* 全职匹配我最爱，最左前缀要遵守；
* 带头大哥不能死，中间兄弟不能断；
* 索引列上少计算，范围之后全失效；
* LIKE百分写最右，覆盖索引不写\*；
* 不等空值还有OR，索引影响要注意；
* VAR引号不可丢， SQL优化有诀窍。

## 批量导入



### insert语句优化；

* **提交前关闭自动提交**
* **尽量使用批量insert语句**
* **可以使用MyISAM存储引擎**

### LOAD DATA INFLIE

LOAD DATA INFLIE；

使用LOAD DATA INFLIE ,比一般的insert语句快20倍

select \* into OUTFILE 'D:\\product.txt' from product\_info

load data INFILE 'D:\\product.txt' into table product\_info