**设计性实验九**

1使用源码查询1中的user表完成如下操作

1. 使用CREATE INDEX语句为name字段创建长度为10的索引index\_name。

CREATE INDEX index\_name ON users(username(10));

1. 使用ALTER TABLE语句在username和age上创建名为index\_u\_a的索引。

ALTER TABLE users ADD INDEX index\_u\_a(username,age);

1. 使用ALTER TABLE语句在addr字段上创建名为index\_addr的全文索引。

ALTER TABLE users ADD FULLTEXT INDEX index\_addr ON users(addr);

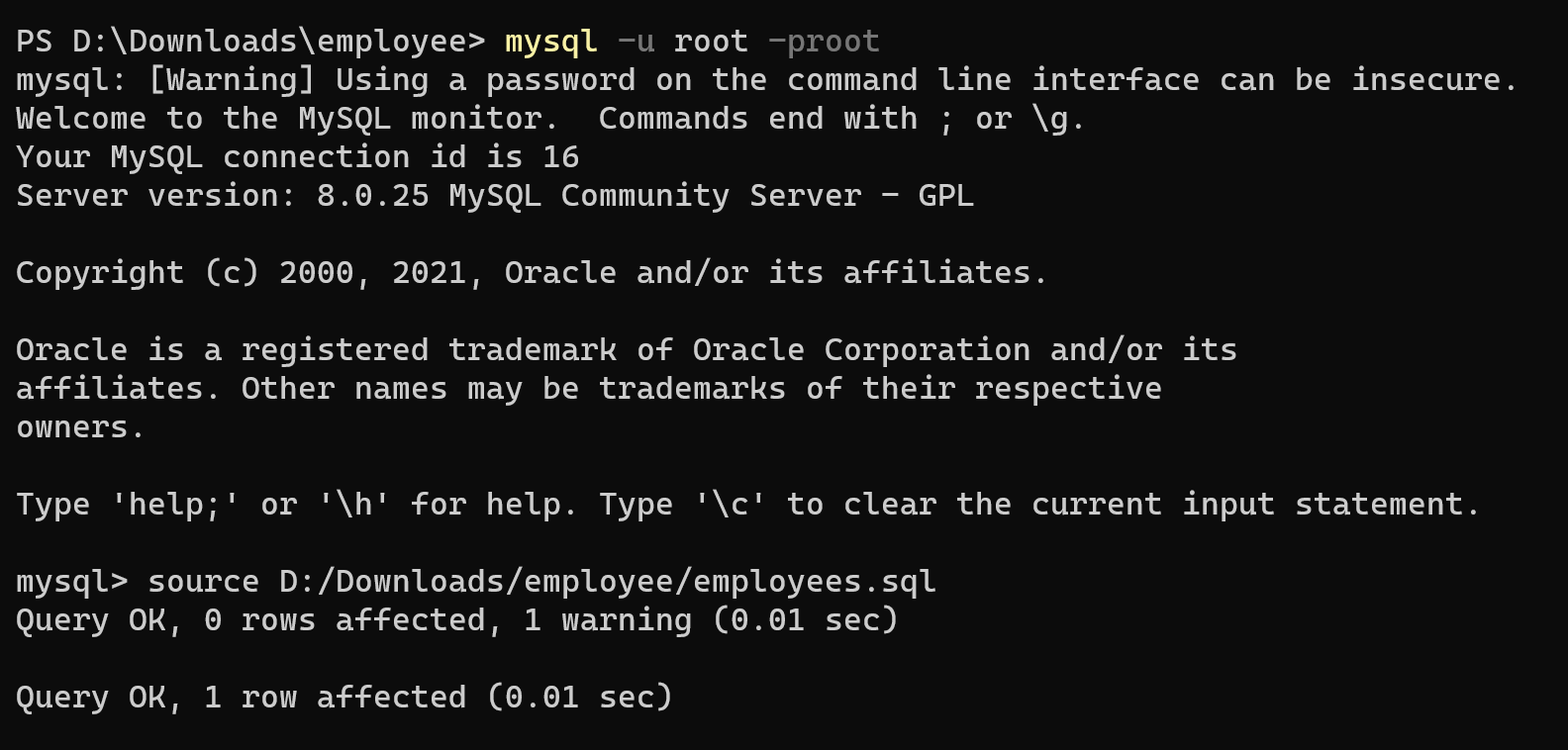
1. 使用DROP语句删除user表name字段上的唯一索引

DROP INDEX index\_name ON users;

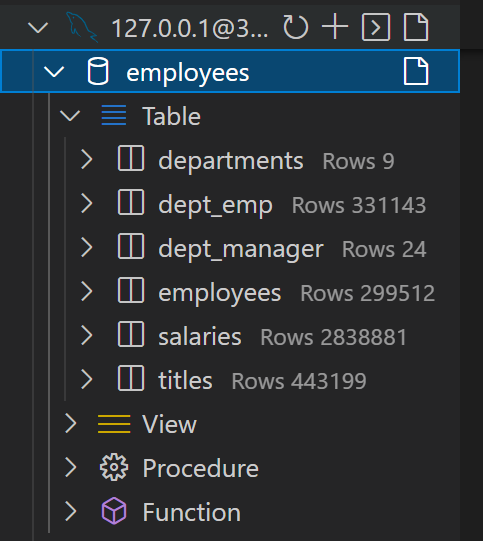
2 验证性实验

1. 导入示范数据库employees

使用指令导入数据库：



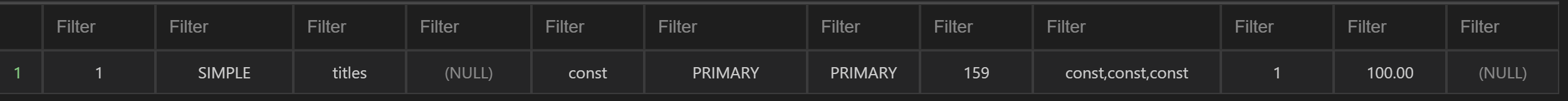
导入成功：



1. 验证联合索引的左匹配原则，并将该原则加以总结。
   1. 左前缀原理：

情况一：全列匹配

EXPLAIN SELECT \* FROM titles WHERE emp\_no='10001' AND title='Senior Engineer' AND from\_date='1986-06-26'\G



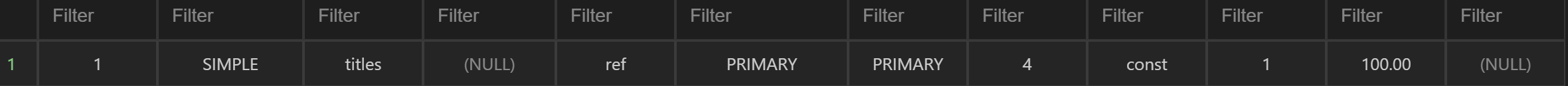
EXPLAIN SELECT \* FROM titles WHERE from\_date='1986-06-26'AND title='Senior Engineer' AND emp\_no='10001' \G



结论：length=(50X3+2)+4+3=159，其中字符变量长度152，整型变量4，日期变量3。在两个例子中使用不同顺序的where条件，但是得到了相同长度的length。在where条件中，不管条件顺序，只要where中出现的列在多列索引中能够从最左开始连贯起来就能使用到多列索引。

情况二 ：最左前缀匹配

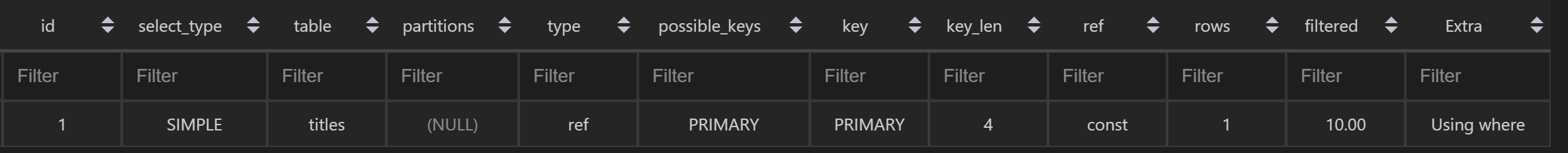
EXPLAIN SELECT \* FROM titles WHERE emp\_no='10001'\G



结论：length=4，只检索了整型变量的长度4，符合最左前缀匹配。

情况三：查询条件用到了索引中列的精确匹配，但中间某个条件未提供

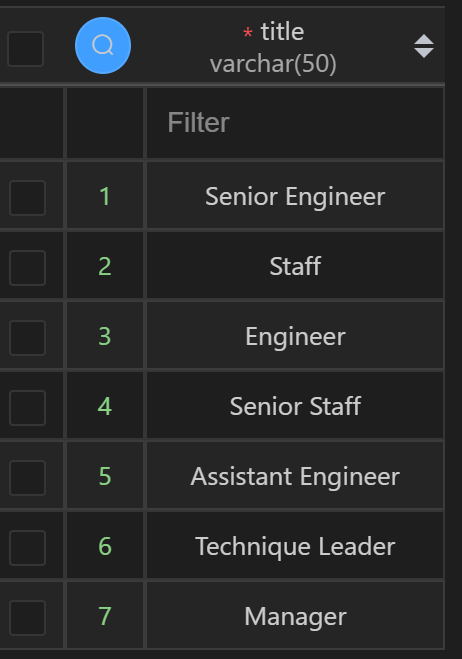
EXPLAIN SELECT \* FROM titles WHERE emp\_no='10001'AND from\_date='1986-06-26'\G



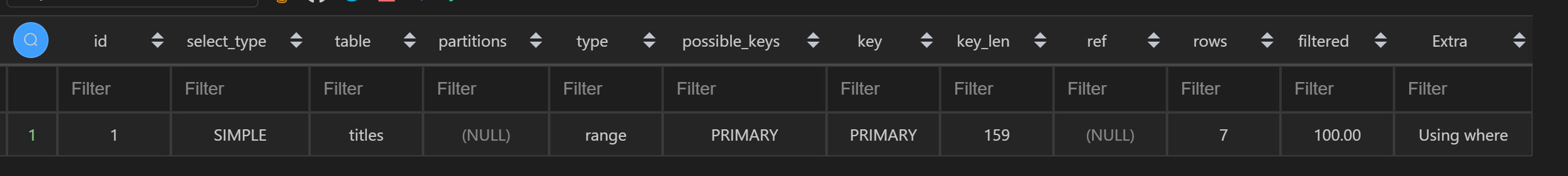
结论：由于中间值为提供，key\_len的长度仅为4，由于中间条件没有提供，多列索引只检索到emp\_no这一列

补坑

SELECT DISTINCT(title) FROM titles;



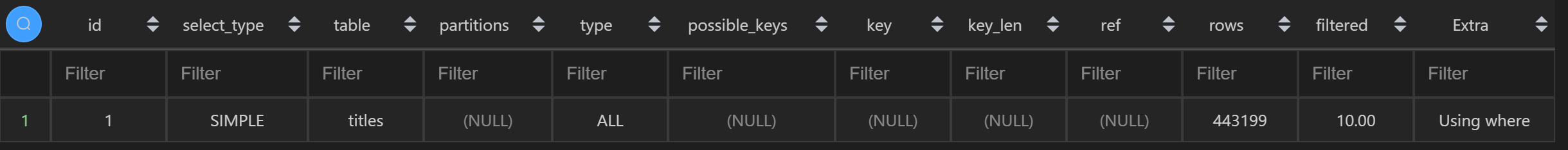
EXPLAIN SELECT \* FROM titles WHERE emp\_no='10001'AND title IN ('Senior Engineer','Staff','Engineer','Senior Staff','Assistant Engineer','Technique Leader','Manager') AND from\_date='1986-06-26'\G



结论：在title项中一共有7条不重复的项。如果在检索时只搜索emp\_no和from\_date会导致多列索引不能全部引用，增加title IN ('Senior Engineer','Staff','Engineer','Senior Staff','Assistant Engineer','Technique Leader','Manager')这一句可以使多列索引全部可用，keylen=159，同时达到了同样的搜索效果。

情况四：查询条件中没有指定索引第一列

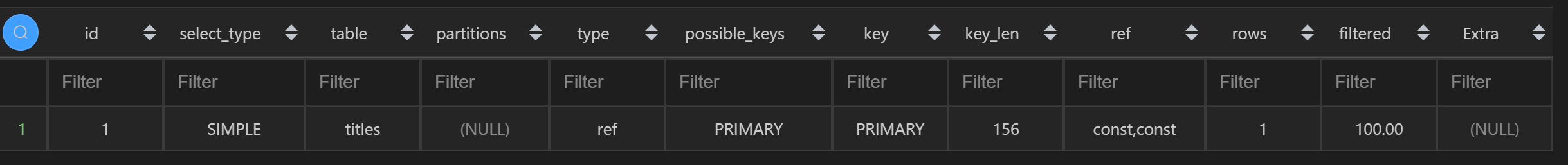
EXPLAIN SELECT \* FROM titles WHERE from\_date='1986-06-26' \G



结论：没有指定索引第一列，所以没有possible\_key可以使用。

情况五：匹配某列的前缀字符串

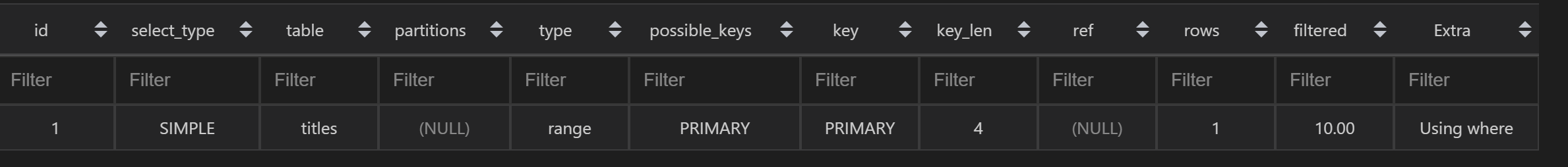
EXPLAIN SELECT \* FROM titles WHERE emp\_no='10001' AND title='Senior%' \G



结论：type为ref，使用非唯一索引扫描或唯一索引前缀扫描，返回单条记录

情况六：范围查询

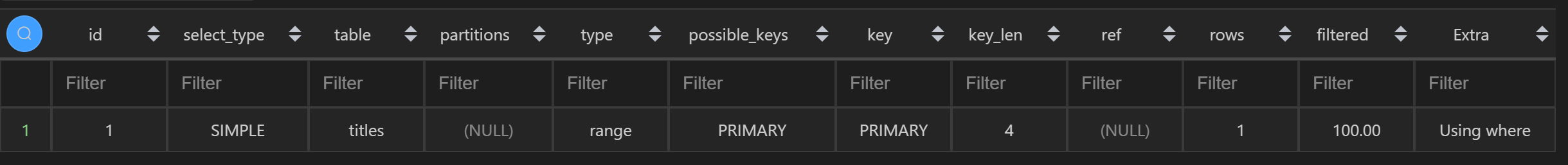
EXPLAIN SELECT \* FROM titles WHERE emp\_no<'10001' AND title='Senior Engineer'\G



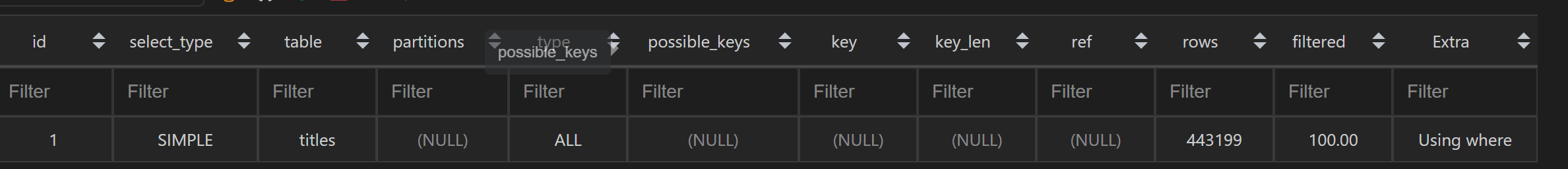
结论：type为range，索引范围扫描

情况七：查询条件中含有函数或表达式

EXPLAIN SELECT \* FROM titles WHERE emp\_no<'10001' AND left(title,6)='Senior'\G



EXPLAIN SELECT \* FROM titles WHERE emp\_no-1='10000'\G



结论：第一种函数使用了索引，keylen=4，说明多列检索只检索了emp\_no。而在第二个函数中，possiblekey为空，说明没有使用key检索。

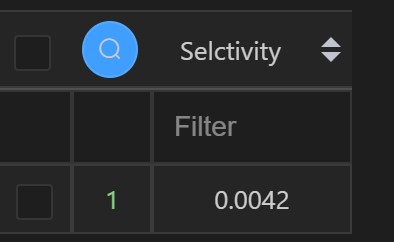
* 1. 索引选择性与前缀索引：

SELECT COUNT(DISTINCT(title))/COUNT(\*) AS Selctivity FROM titles;



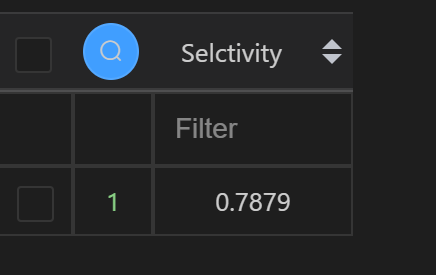
Title的索引选择性接近0

SELECT COUNT(DISTINCT(first\_name))/COUNT(\*) AS Selctivity FROM employees;



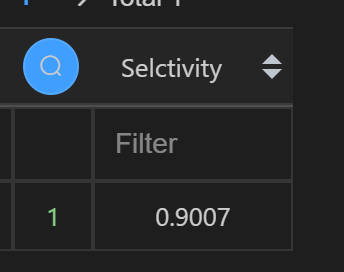
First\_name的索引选择性为0.0042

SELECT COUNT(DISTINCT(CONCAT(first\_name,left(last\_name,3))))/COUNT(\*) AS Selctivity FROM employees;



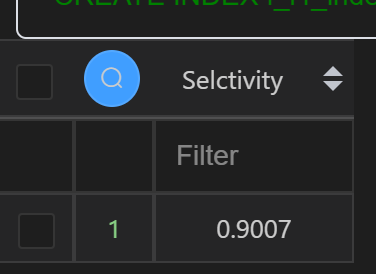
Firstname加lastname的前三个字符，选择性0.7879

SELECT COUNT(DISTINCT(CONCAT(first\_name,left(last\_name,4))))/COUNT(\*) AS Selctivity FROM employees;



Firstname加lastname的前三个字符，选择性0.9007

CREATE INDEX f\_l4\_index ON employees(first\_name,last\_name(4));



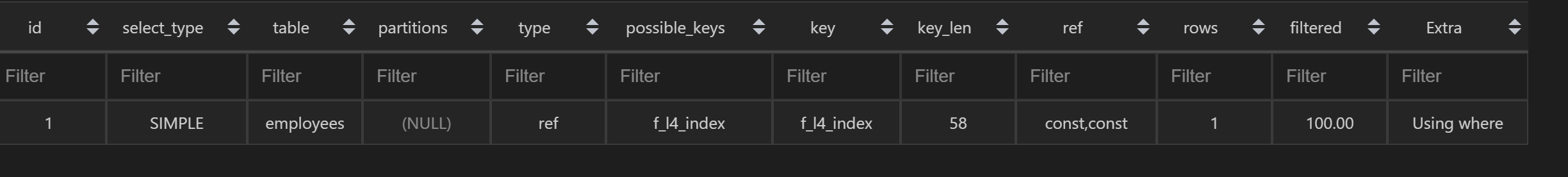
Firstname加lastname的前四个字符，选择性0.9007

CREATE INDEX f\_l\_index ON employees(first\_name,last\_name);

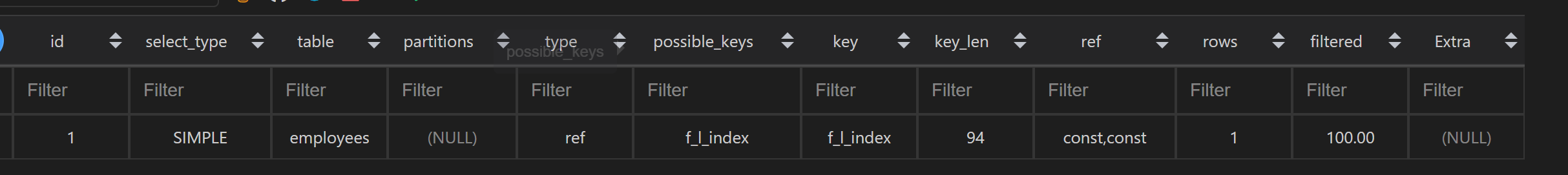


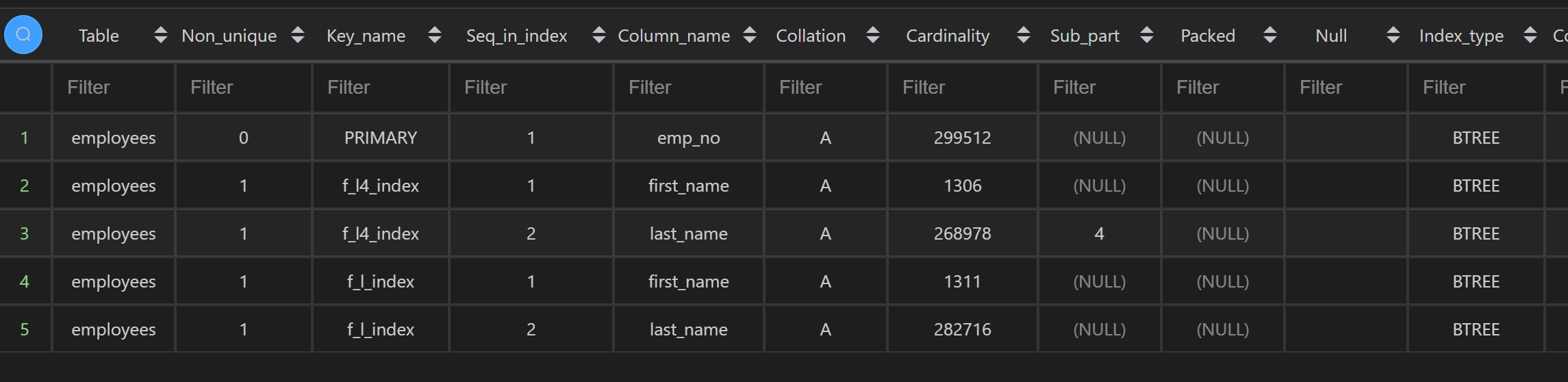
Firstname加lastname，选择性0.9007。与选择latname前四个字符相同。

EXPLAIN SELECT \* FROM employees USE INDEX FOR JOIN (f\_l4\_index)WHERE first\_name='Eric' AND last\_name='Anido'\G



EXPLAIN SELECT \* FROM employees USE INDEX FOR JOIN (f\_l\_index)WHERE first\_name='Eric' AND last\_name='Anido'\G





f\_I4\_index使用lastname的前四个字符，f\_I4\_index使用lastname全部。虽然他们的选择性相同，但是keylen上显然f\_I4\_index更短。得到结论，选用合适的字段部分进行索引可以增加索引效率。