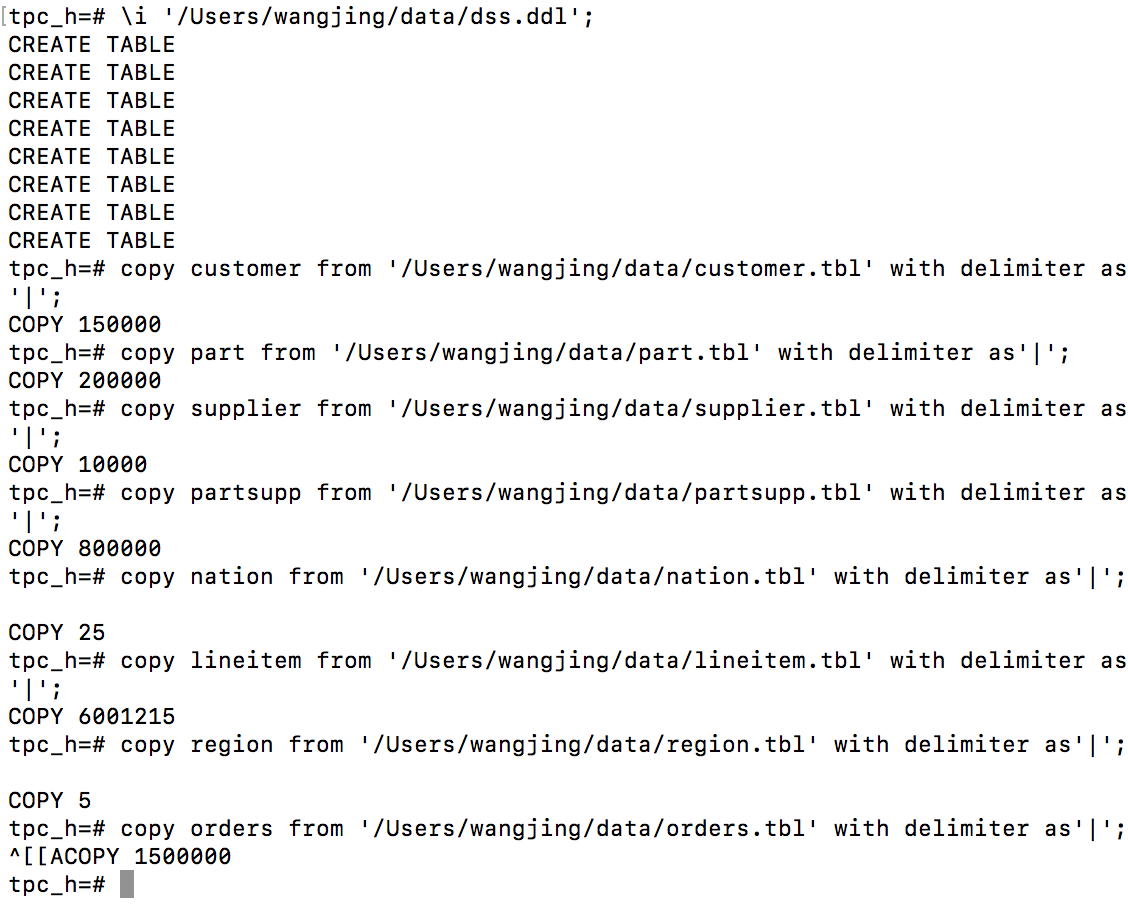
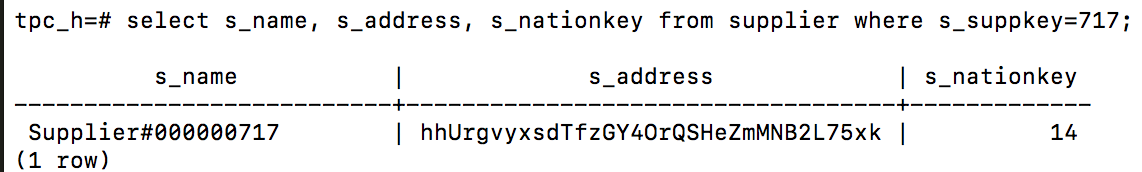
Task1:

创建了一个

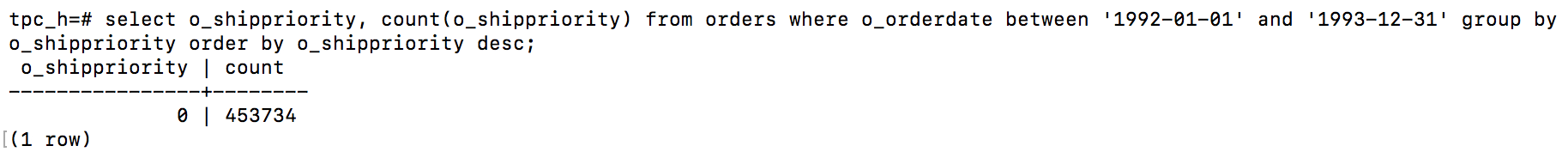


Task2:

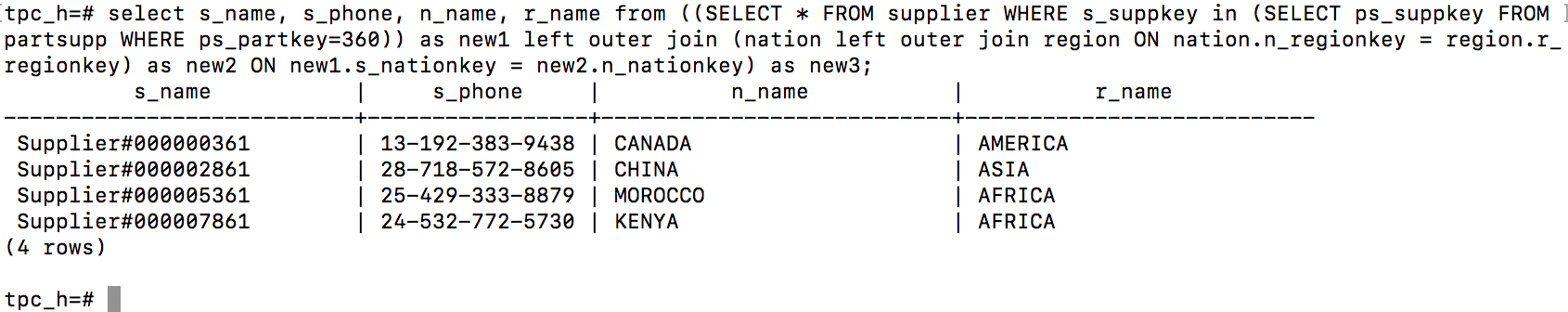
1. 展示suppkey为717的supplier的name，address和nationkey，那么就要在表supplier中进行查找。



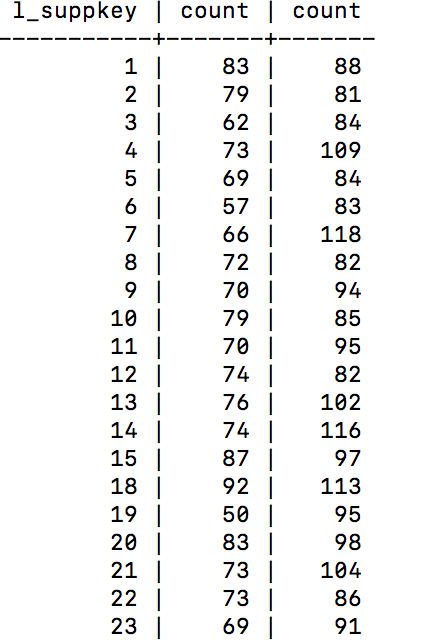
1. 对于每个ship priority，计算其在1992到1993之间的总数，并降序排列展示。操作过程中发现，ship priority都为0。



1. 找到为partkey为part的supplier，然后展示它们的name，phone，nation和region。

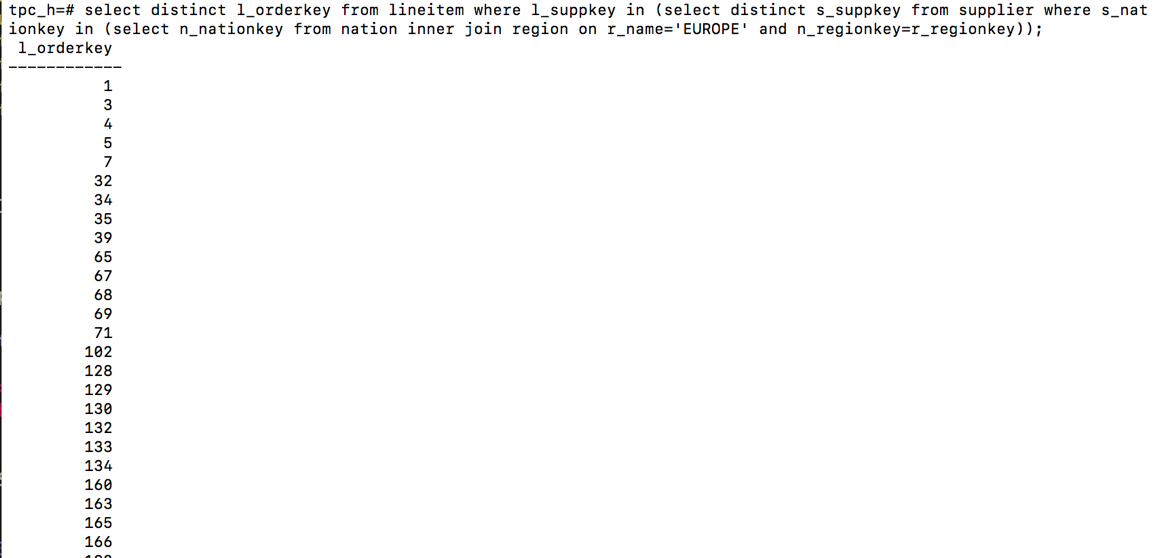


1. 找到那些1993年供应比1992年更多part的supplier，然后展示它们的及其在1992年和1993年分别的供应量。

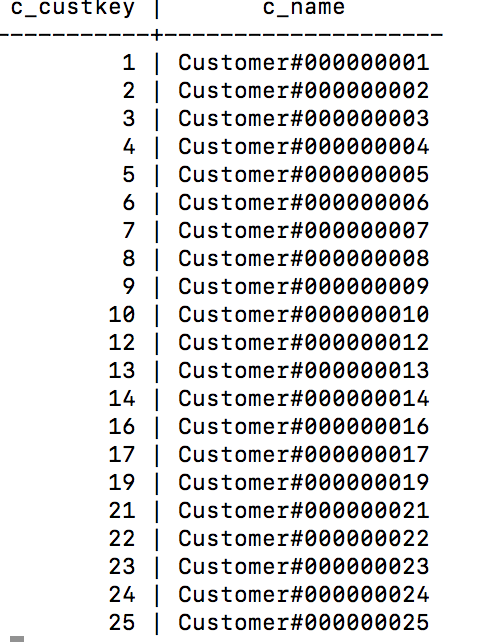


左边的count是1992年的供应量，右边的count是1993年的供应量。

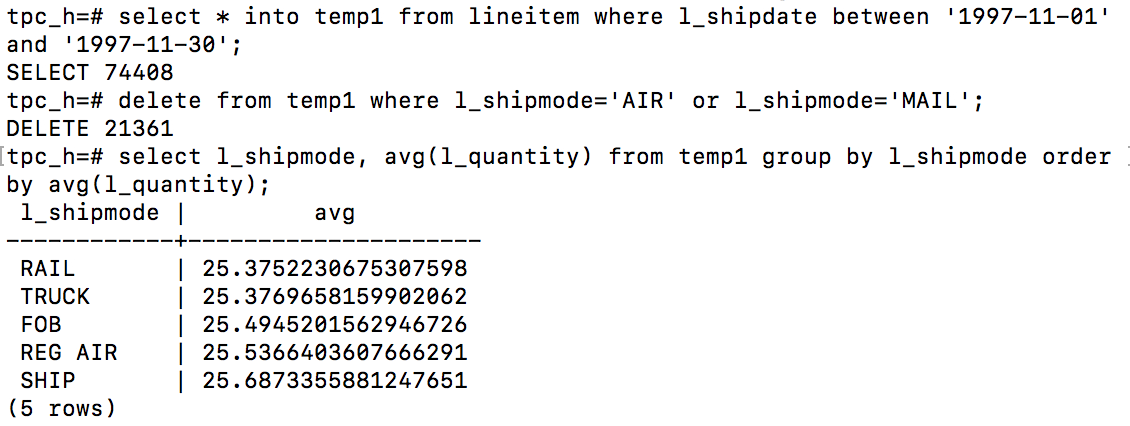
1. 选出来自欧洲地区的supplier参与供应的orders



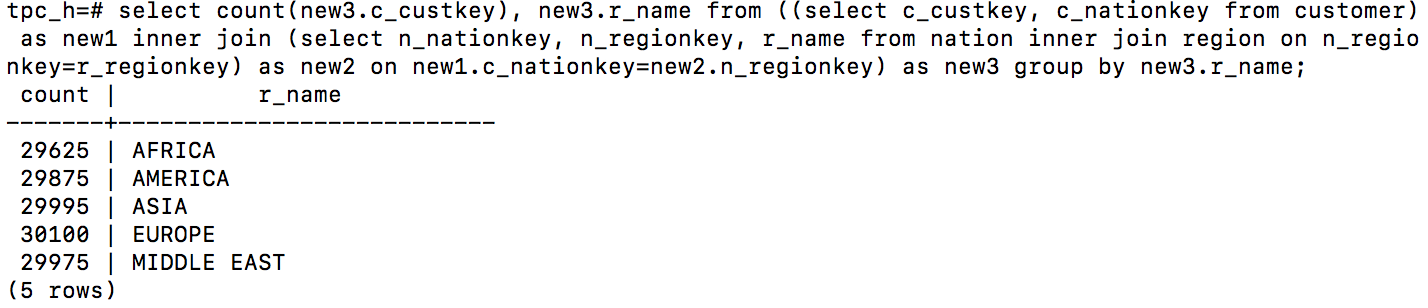
1. 选出不是来自欧洲的customer



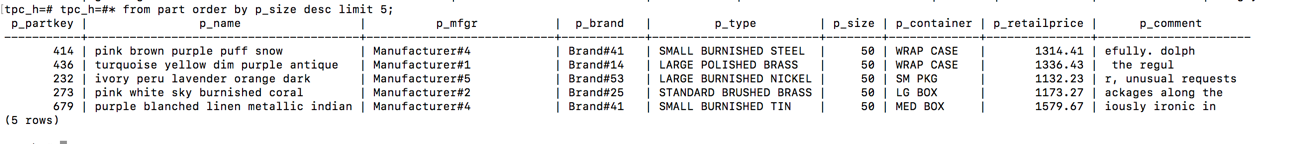
1. 选出1997年11月订购的各类非‘AIR’和‘MAIL’型的船的平均质量，并且以生序排列显示。



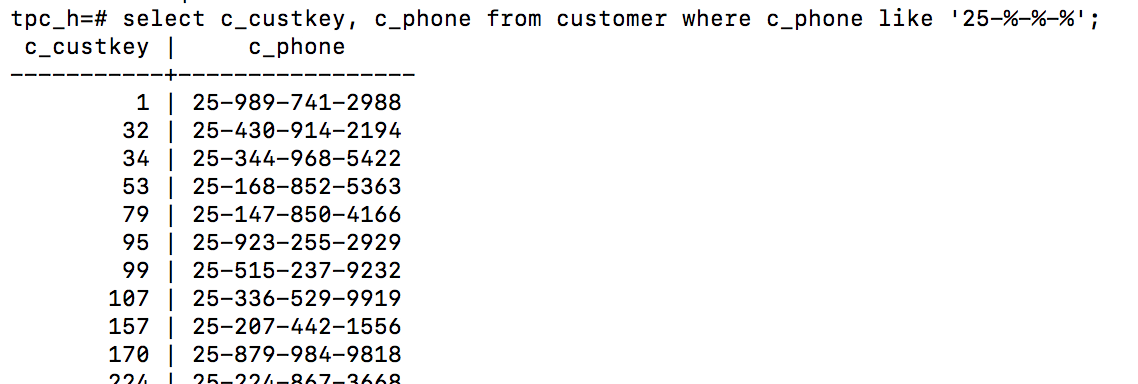
1. 展示每个地区的customer的数量。



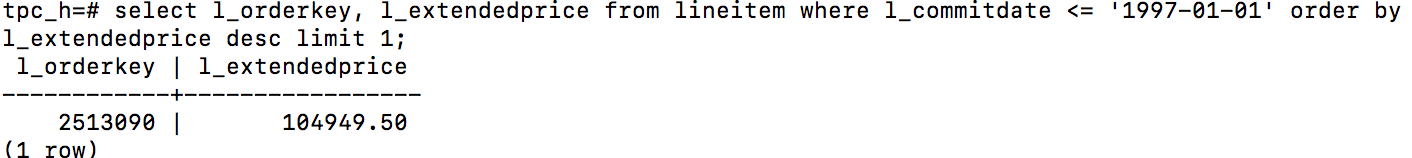
1. 展示最上端的5个size最大的part的所有信息。



1. 找到电话号码格式为25-XXX-XXX-XXX的customer



1. 找到有最大extend price的并且在1997年之前订购的part，并且展示这个最大的extend price。

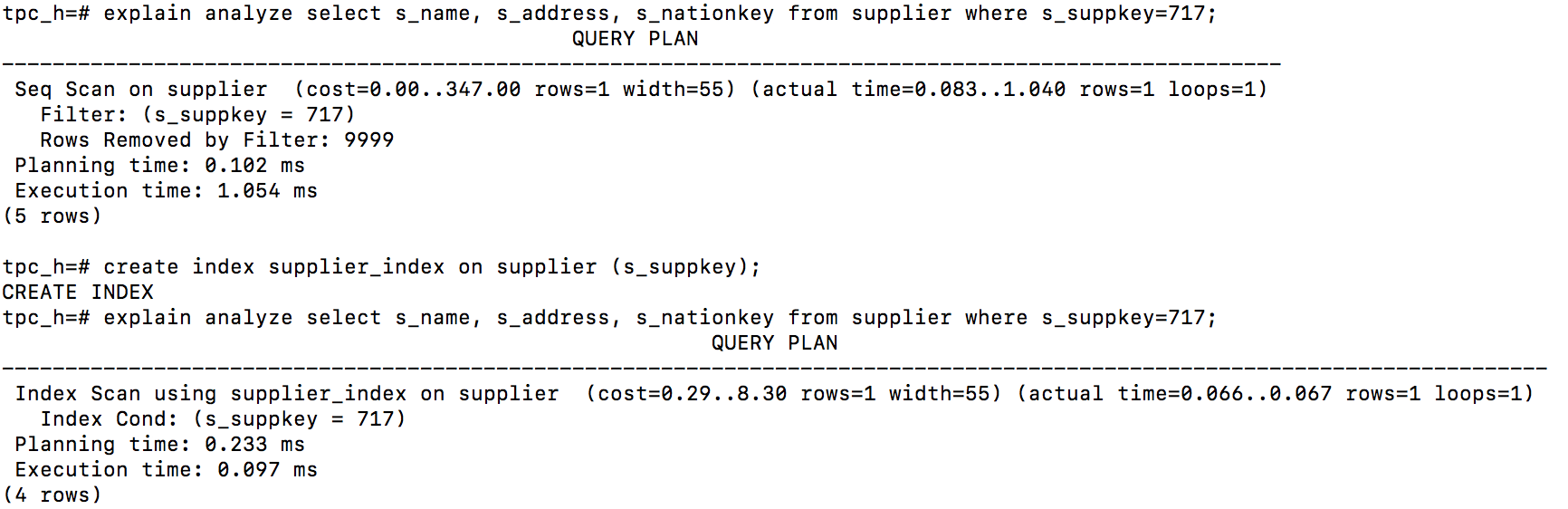


Task3:

首先是默认的b-tree索引：

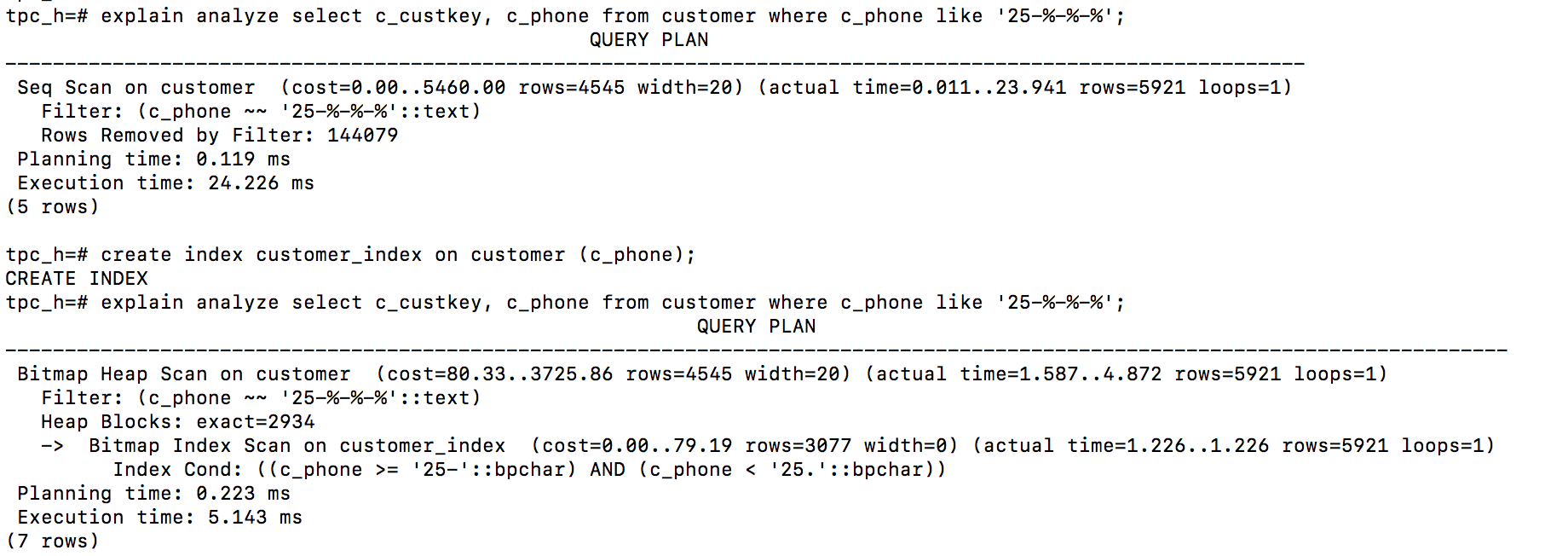
通过建立索引，然后重新执行a,j,k题的语句，并分析执行时间。

(a)题的结果，建立索引前和建立索引后：



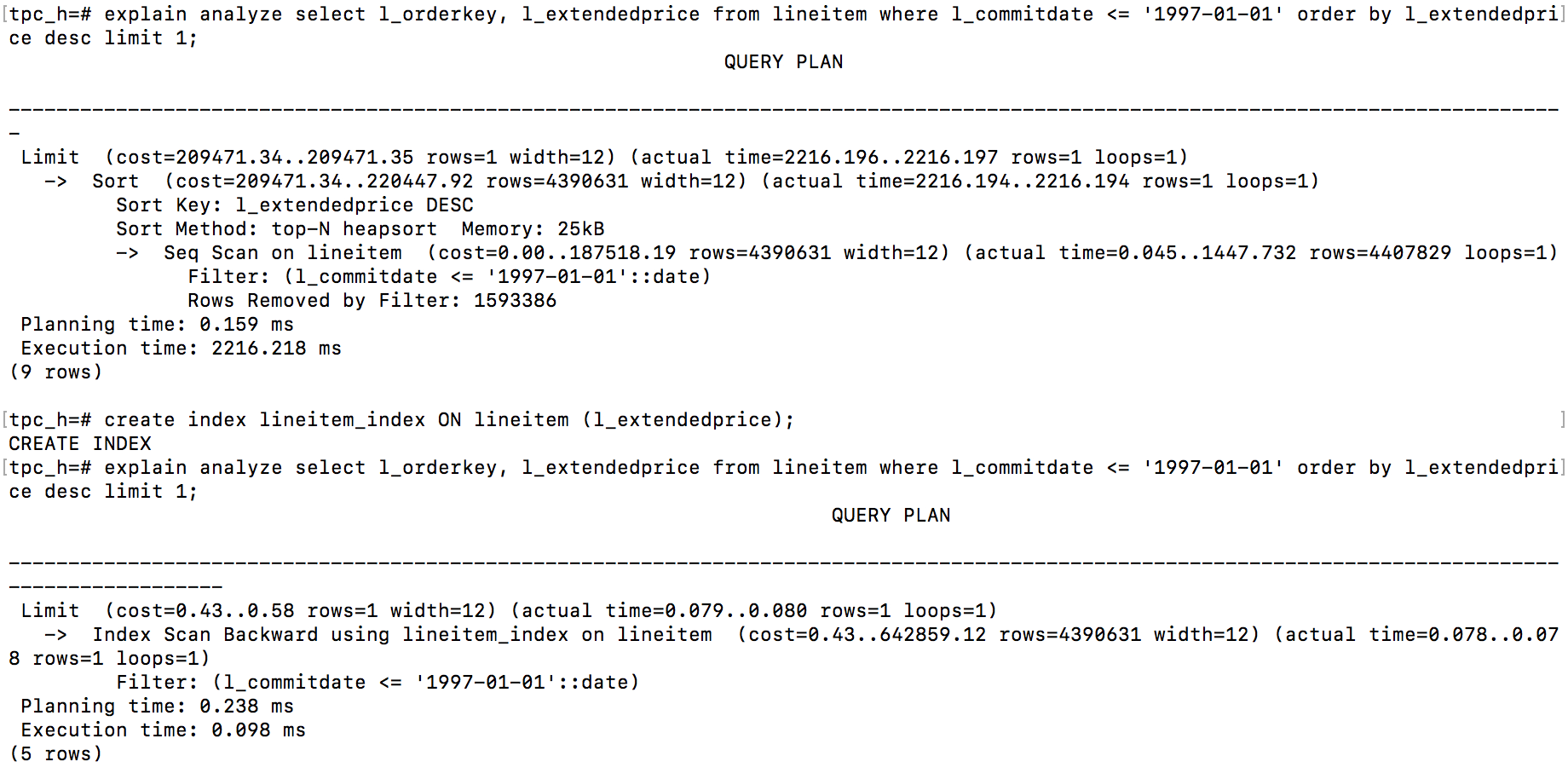
可以看到建立前execution time为1.054ms，建立后为0.097ms，明显缩短了时间。

(j)题的结果，建立索引前和建立索引后：



可以明显的看出execution time的区别，建立索引后的的增速。

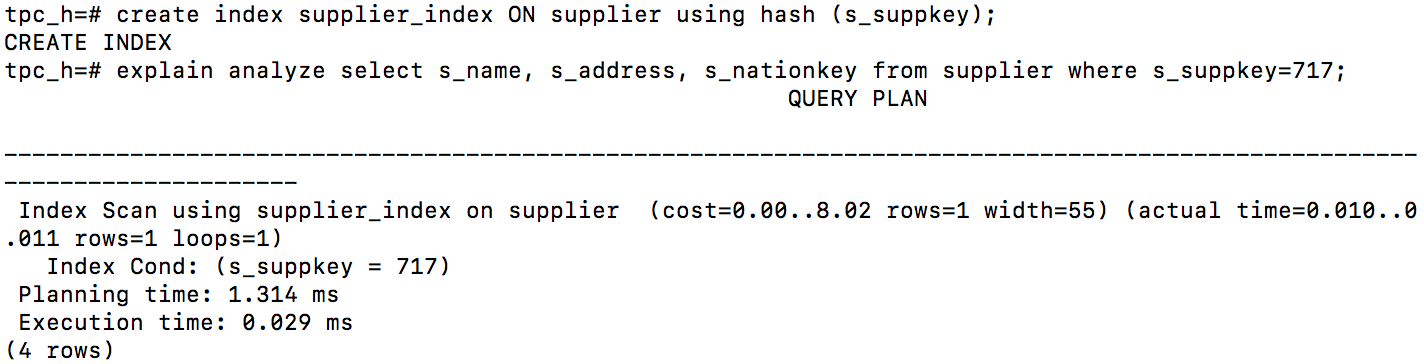
(k)题的结果，建立索引前和建立索引后：



这个结果的增速就非常明显了

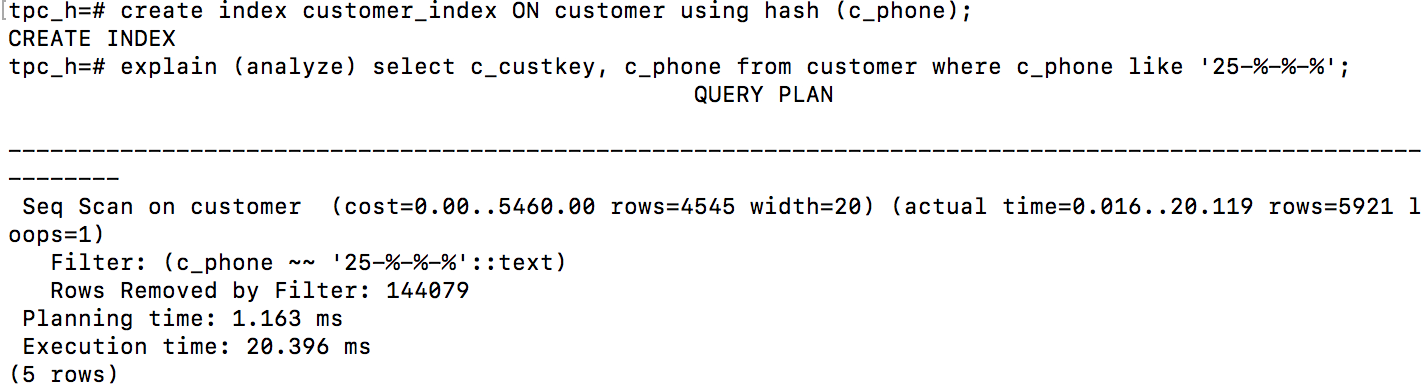
而使用hash索引：

(a)的情况：



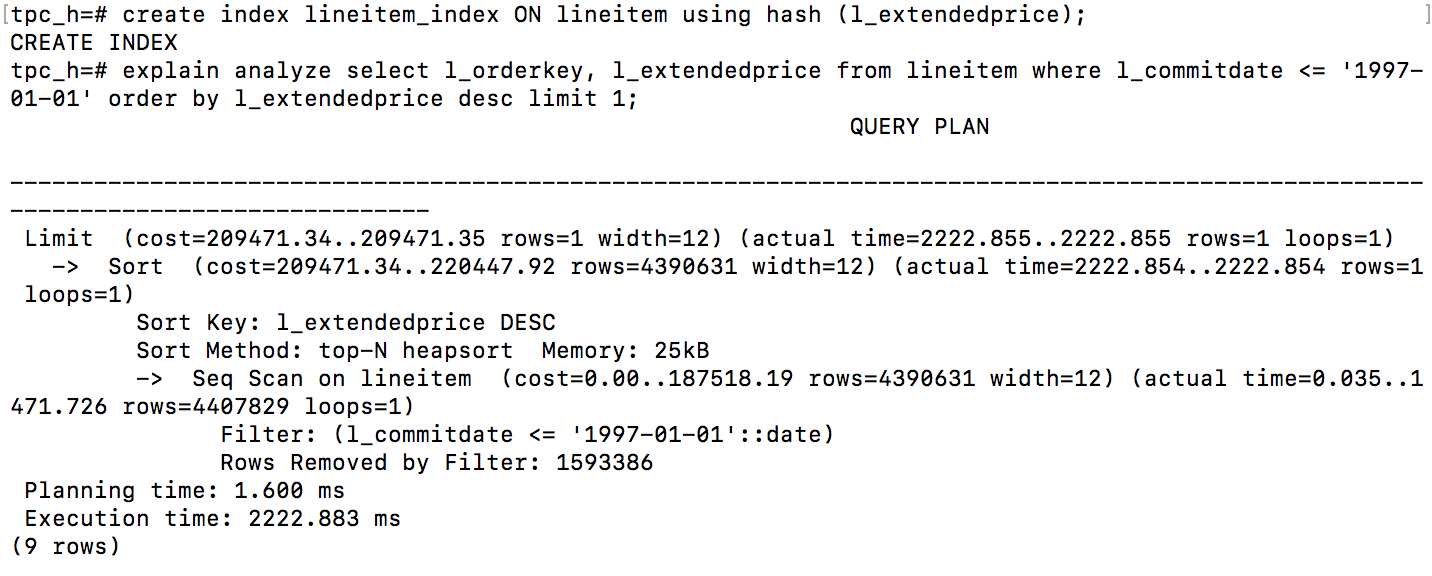
增速比用b-tree的更多。

(j)的情况：



不如b-tree

(k)的情况：



不如b-tree

综上可以看出，当需要查找某个具体的数据项时，hash索引的速度比b-tree索引的效率要高；而需要用到排列和大范围匹配的时候，b-tree的效率会比hash索引的高。