北京邮电大学

实验报告

课程名称：数据库系统原理

实验名称：数据查询分析优化实验

计算机系2015211312班 姓名刘佳鑫

计算机系2015211312班 姓名万诗婕

教师吴起凡 成绩\_\_\_\_\_\_\_\_\_

2018年5月31日

### 实验目的

* 1. 在SQL Server Management Studio平台下，通过观察Select/Insert/delete/update等SQL语句的查询执行计划，分析查询执行计划中连接、选择、投影等关系代数操作的实现方式及其执行成本。熟悉了解SQL SERVER数据库中查询优化的使用，理解数据库查询优化的基本概念。
  2. 掌握利用SQL Server Management Studio提供的机制，分析对比形式不同、执行结果等价的不同SQL语句的查询执行计划的执行成本和执行时间差异。
  3. 熟悉了解视图和with临时视图的创建，观察视图查询、with临时视图查询的执行计划。
  4. 参照文档“数据库物理设计及查询优化”中SQL语句查询优化相关内容，在多种情况下，对比实现方式不同但查询结果相同的等价SQL语句在执行计划和成本方面的差异，加深对查询优化的理解，进行书写优化SQL语句的初步训练，提高编写高效SQL语句的能力。涉及以下几方面：

1）复合索引左前缀；

2）多表连接属性上建立索引；

3）索引对小表查询的作用；

4）查询条件中函数对索引的影响；

5）多表嵌入式SQL查询；

6）where查询条件中复合查询条件OR对索引的影响；

7）聚集运算中索引设计；

8）Select子句中有无distinct的区别；

9）union、union all的区别；

10）from子句中多余关系的影响。

### 实验环境

采用Microsoft SQL Server数据库管理系统作为实验平台。

### 实验内容

1. SQLServer执行计划的查看与分析
2. 观察视图查询、with临时视图查询的执行计划
3. 优化SQL语句

### 实验步骤

### 实验8.1 SQLServer执行计划的查看与分析

#### 实验要求

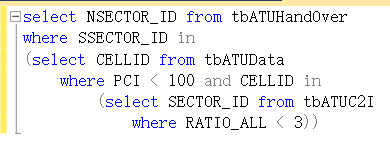
在SQL Server Management Studio平台下，按照下述实验步骤，编写相应的SQL语句，观察Select/Insert/delete/update等SQL语句的查询执行计划，分析查询执行计划中连接、选择、投影等关系代数操作的实现方式，观察分析查询执行计划中关系代数操作的执行成本。

#### 实验步骤

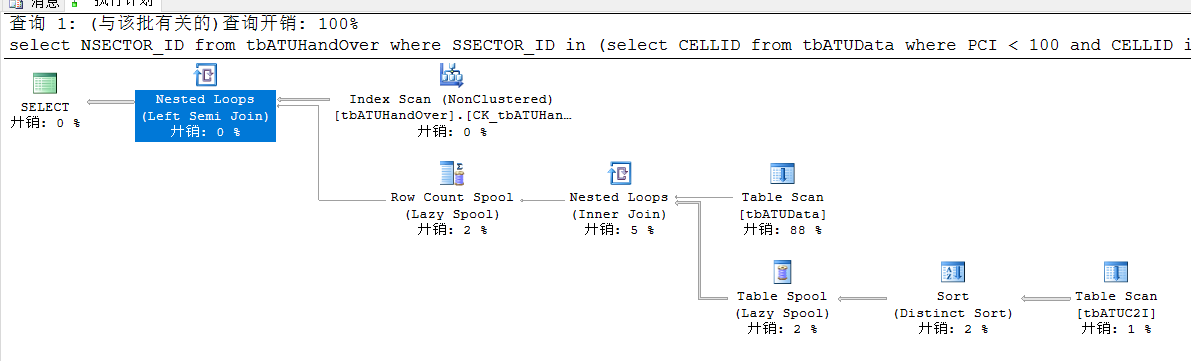
1. 编写SQL语句完成下面查询；

1.1 查询要求

查询tbATUData中PCI小于100且在tbATUC2I中干扰强度小于3的主小区的切换目标小区。



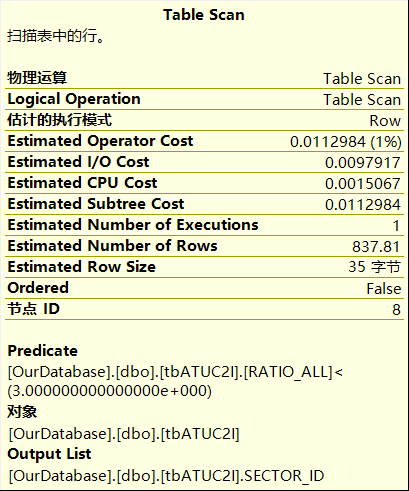
1.2 查看执行计划

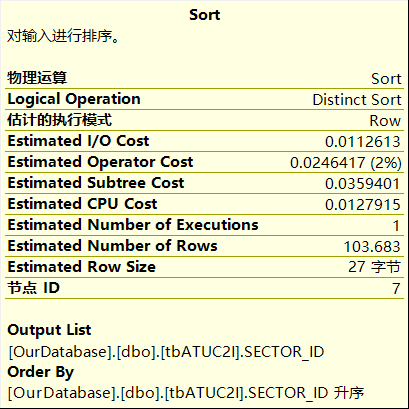
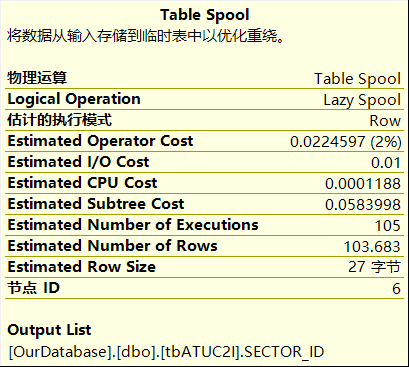


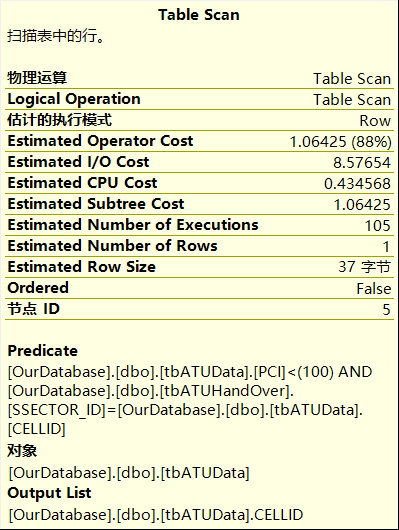
分析

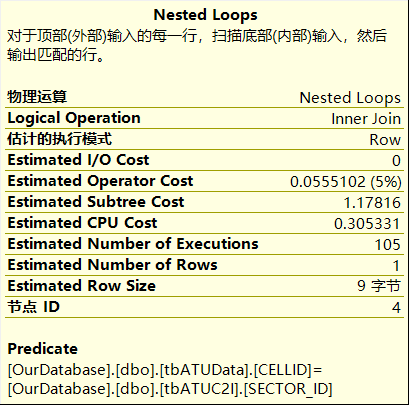
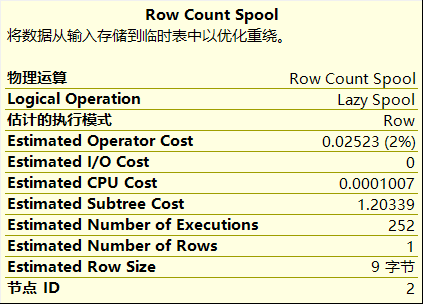
执行计划应该从右往左看，箭头的粗细表示每一步产生的数据量的大小，每一步操作下面有其对应的开销，理论上，通过优化开销最大的部分可以优化整个查询。上述主要通过以下几个步骤完成查询：

1. 在tbATUC2I的全表扫描（开销：6%），从而筛选出符合条件的SECTOR\_ID（RATIO\_ALL<3）；



1. 对tbATUC2I选出SECTOR\_ID的结果进行升序排列（开销2%）
2. 将tbATUC2I选出的结果放入一个临时表中，以便优化环绕（开销2%）；
3. 对tbATUData进行全表扫描（开销88%），输出其CELLID；

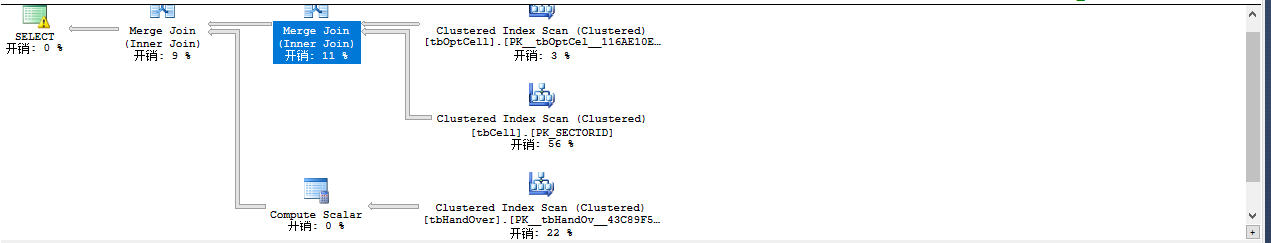


1. 将tbATUC2I的查询结果和tbATUData的结果进行连接操作（开销5%）。
2. 将连接的结果输入到临时表中已优化重绕（开销2%）

通过上面的分析可以看出对于tbATUData表格的扫描占据了最大的开销，所以我们可以通过增加索引来减少对于tbATUData表格扫描的开销

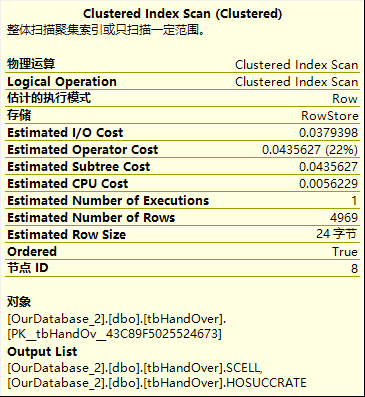
2.1 查询要求

查询tbHandOver中小区类型为“优化区”且设备厂家为华为的切换源小区和切换成功率。

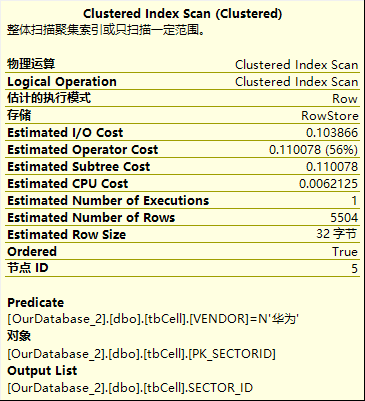
2.2 查看执行计划

分析：执行计划应该从右往左看，箭头的粗细表示每一步产生的数据量的大小，每一步操作下面有其对应的开销，理论上，通过优化开销最大的部分可以优化整个查询。上述主要通过以下几个步骤完成查询：

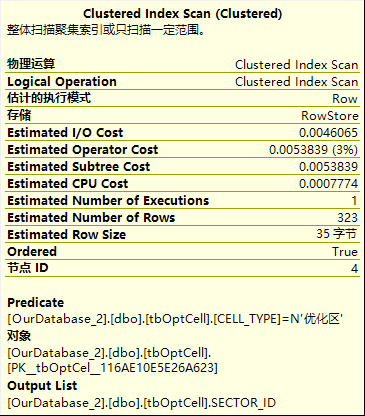
（1） 聚集索引扫描，聚集索引为SCELL和NCELL，选出了tbHandOver的SCELL和HOSUCCRATE的属性。开销为22%



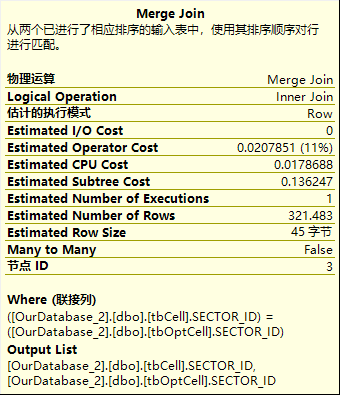
（2） 聚集索引扫描，选出了tbCell的SECTOR\_ID属性。选出了VENDOR=华为的行。开销为56%



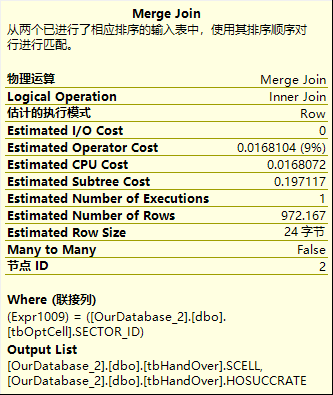
（3）聚集索引扫描，选出了tbOptCell中SECTOR\_ID属性，选出了CELL\_TYPE=优化区的行。开销为3%



（4）将tbCell和tbOptCell连接起来，条件是tbCell.SECTOR\_ID = tbOptCell.SECTOR\_ID，开销为11%



（5）将tbHandOver和刚才连接后的表再做连接，条件为SCELL和tbOptCell的SECTOR\_ID相同。开销为9%。之后便可选出。



此次查询中扫描tbCell的开销最大。

### 实验8.2 观察视图查询、with临时视图查询的执行计划

#### 实验要求

从实验数据库中选取一张表在上面建立视图和with临时视图，执行一个查询，观察其执行计划。

#### 实验步骤

1. 选取数据表的一张表（示例中的表除外），分别在上面建立视图和with临时视图；
2. 分别在两个视图上执行同一个查询语句，将它们同时提交给DBMS；
3. 查看执行计划并分析。

#### 示例

1. 分别在tbCell上建立视图和临时视图

建立视图：

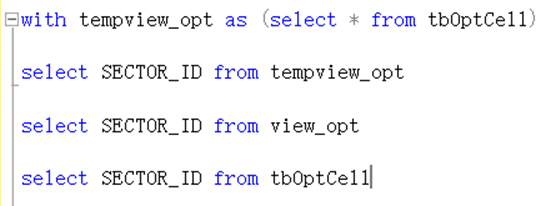
create view view\_opt as select \* from tbOptCell

建立临时视图：

with tempview\_opt as (select \* from tbOptCell)

select SECTOR\_ID from tempview\_opt

1. 查询



1. 查看执行计划



说明：通过执行计划发现在视图、临时视图和原表上的查询并没有太大的差异，但这并不代表说视图对性能无影响。从网上查询得知，对于MySQL来说，其对视图有不同的代码库，所以临时视图会取得更好的性能。

### 实验8.3 优化SQL语句

结合文档“数据库物理设计及查询优化”中相关内容，通过对以下各种不同情况下查询语句的执行情况的对比分析，巩固加深查询优化的理解，并进行书写优化SQL语句的初步训练，提高编写高效SQL语句进行数据查询的能力。

### 8.3.1 复合索引左前缀

##### 实验要求

在做这部分实验时，要求使用tbATUData表,按照下述步骤完成实验内容。对访问相同的两张表且查询需求完全一样的2条select语句实现两个要求（在tbATUData上创建包含至少三个索引的组合索引）：

1. 比较有最左前缀索引和无索引的2条select语句的执行结果和执行计划；
2. 比较组合索引中其他索引（不包含最左前缀索引）和无索引的2条select语句的执行计划。

分别判断2条语句的执行结果是否一样，对比执行效果和执行速度，解释执行时间出现差异的原因。

##### 实验步骤

1. 创建tbATUData的备份表tbATUDataNew；

select \* into tbATUDataNew from tbATUData

1. 在tbATUDatanew创建组合索引（至少包含三个索引）。索引属性建议：可以在Longitude、Latitude、CellID上设计组合索引。

create index llc on tbATUdataNew(LONGITUDE,LATITUDE,CELLID)

1. 比较有最左前缀索引和无索引的2条select语句的执行结果和执行计划

编写select语句，使用最左前缀索引访问tbATUDatanew；编写select语句，不使用索引方式，访问tbATUData，将2条select语句，同时提交给DBMS执行后，从2个执行结果窗口中，判断执行的结果是否一致，并观察各自执行效果、查询执行计划、时间对比。

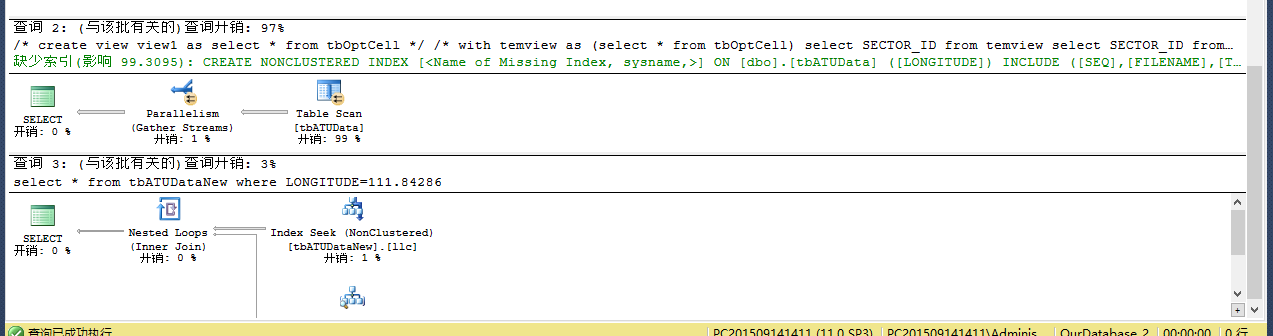
3.1.1 使用最左索引

select \* from tbATUData

where LONGITUDE=111.84286

select \* from tbATUDataNew

where LONGITUDE=111.84286



说明：有索引的性能更优，最左索引可以起到作用，所以能大大优化性能。

* + 1. 在比较组合索引中其他索引（不包含最左前缀索引）和无索引的2条select语句的执行计划

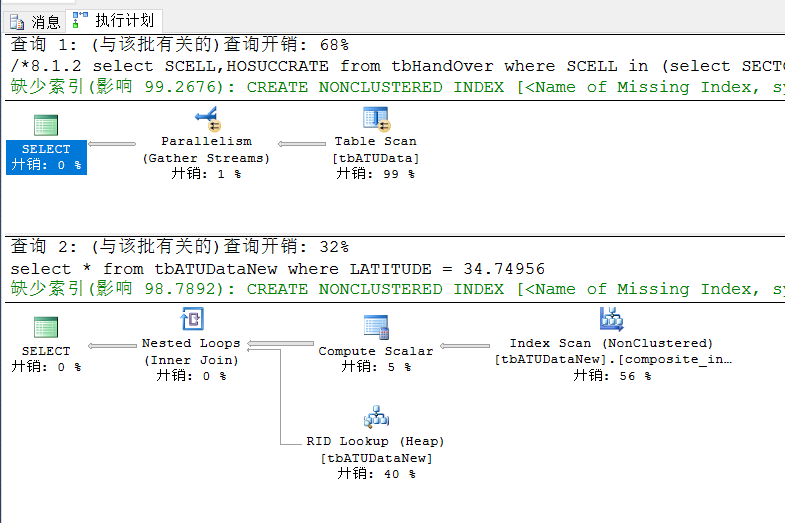
编写select语句，使用不含最左前缀的索引访问tbATUDatanew；编写select语句，不使用索引方式，访问tbATUData，将2条select语句，同时提交给DBMS执行后，从2个执行结果窗口中，判断执行的结果是否一致，并观察各自执行效果、查询执行计划、时间对比。

无索引

select \* from tbATUData where LATITUDE = 34.74956

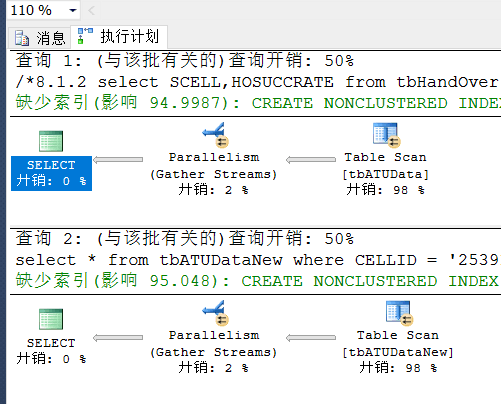
用组合索引中的第二个索引

select \* from tbATUDataNew where LATITUDE = 34.74956



说明：此刻发现非左索引也起到了作用，实现了性能优化。

此时我们使用CELLID作为where的条件：

select \* from tbATUData where CELLID = '253910-0'  
  
select \* from tbATUDataNew where CELLID = '253910-0'

发现执行计划的开销是50%和50%，此时的非最左索引没有起到作用。

之前的问题猜想是因为数据中longitude和latitude数据有一致性，所以会对结果有影响。

### 8.3.2 多表连接操作，在连接属性上建立索引

##### 实验要求

在做这部分实验时，要求使用tbATUData和tbAdjCell表,按照下述步骤完成实验内容。对访问相同的两张表且查询需求完全一样的2条select语句实现两个要求之一（在连接属性上建立索引）：

1. 在tbATUDatanew表的CellID上设计索引，列举在tbAdjCell中找到此小区的主小区频点。

创建索引

create index atu\_index on tbATUDataNew(CELLID)

有索引：

select S\_EARFCN

from tbAdjCell ,tbATUDataNew

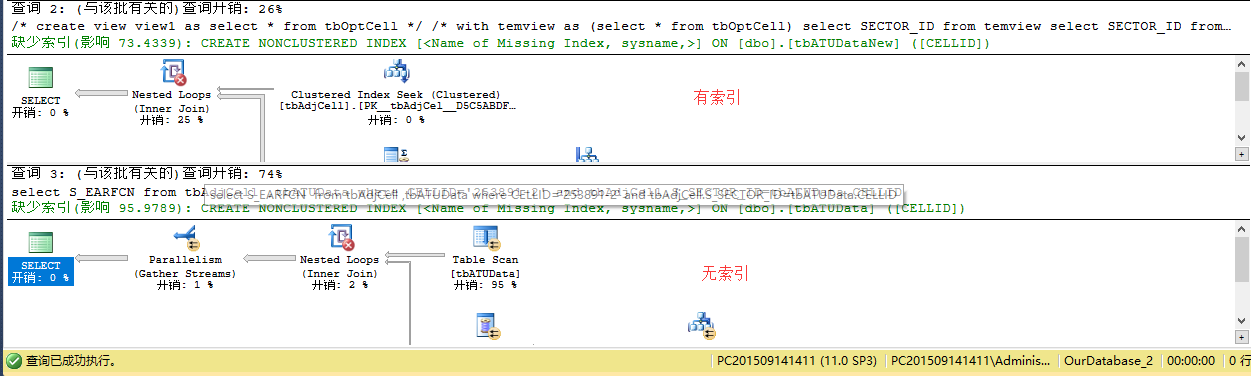
where tbAdjCell.S\_SECTOR\_ID=tbATUDataNew.CELLID

无索引

select S\_EARFCN

from tbAdjCell ,tbATUData

where tbAdjCell.S\_SECTOR\_ID=tbATUData.CELLID



说明：在连接属性上建立索引可以优化性能。

### 8.3.3 索引对小表查询的作用

##### 实验要求

在做此部分实验时要求使用元组数目少、表占用空间少的tbOptCell表，按照下述步骤完成实验内容。在SECTOR\_ID上创建索引，执行一个查询，查看执行计划观察是否用到索引，如果没有则强制使用并与不强制使用做比较。

##### 实验步骤

1. 创建tbOptCell的备份表tbOptCellnew；

select \* into tbOptCellNew from tbOptCell

1. 在tbOptCellnew上创建索引；

create index opt\_index on tbOptCellNew(SECTOR\_ID)

1. 在tbOptCellnew上新建查询，观察是否用到索引；

select \* from tbOptCellNew

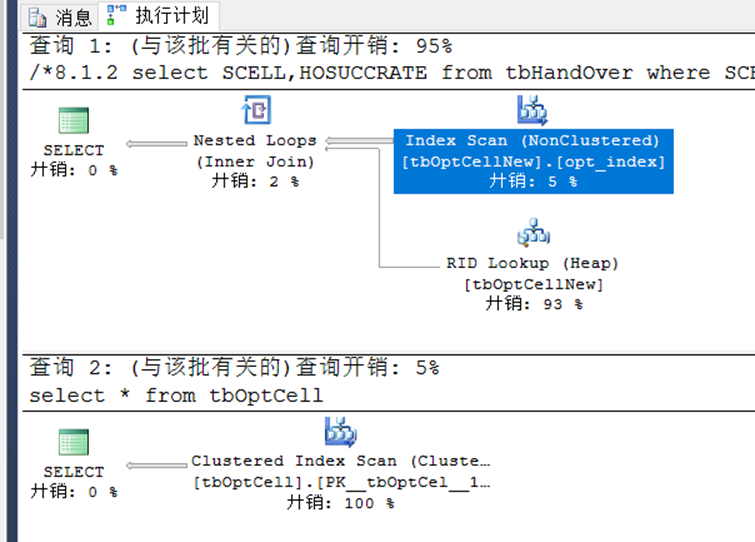


说明：没有使用到索引。

1. 编写select语句，访问tbOptCell，强制使用索引访问tbOptCellnew，将2条select语句同时提交给DBMS执行后，从2个执行结果窗口中，判断执行的结果是否一致，并观察各自执行效果、查询执行计划、时间对比。

强制使用索引：select \* from tbOptCellNew with (index=opt\_index)

强制不使用索引：select \* from tbOptCell



说明：在数据较少的表时，强制使用索引反而会增加开销。所以对于数据较少的表时，建议不使用索引。

### 8.3.4 查询条件中函数对索引的影响

##### 实验要求

针对同一张表，编写2条查询需求完全一样的等价的select语句，一条语句在where子句查询条件中使用了函数，另外一条语句则没有使用函数。

##### 实验步骤

1. 创建备份表并在备份表上创建索引；
2. 编写select语句，完成实验要求中的查询，对备份表同时提交2条等价的查询语句；
3. 查看执行计划，比较两个语句的执行结果和执行速度，观察函数对索引的影响。

（1）在路测ATU数据表tbATUData中查询与经度为111.82841且小区标识为253903-0的小区A周边一定范围内的全部小区B，周边一定范围定义为：

小区B所在基站的经纬度与小区A所在基站的纬度差距满足：

ABS(小区A所在基站纬度-小区A所在基站纬度)<=0.001

使用函数：

select distinct B.CELLID

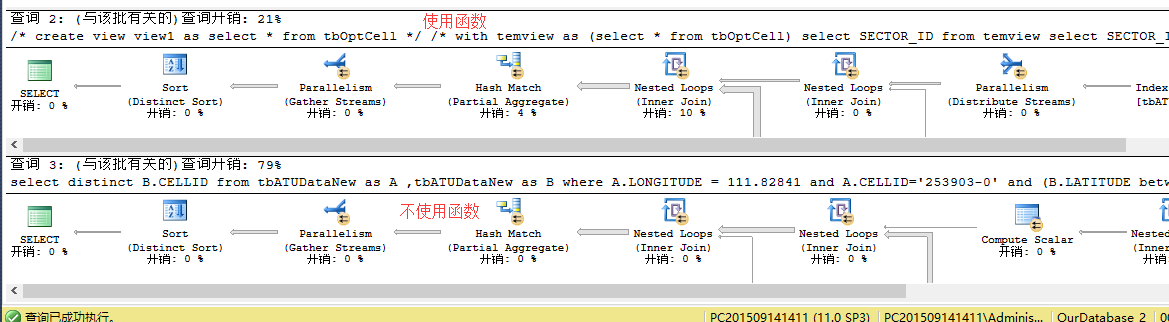
from tbATUDataNew as A ,tbATUDataNew as B

where A.LONGITUDE = 111.82841 and A.CELLID='253903-0' and ABS(A.LATITUDE-B.LATITUDE)<=0.001

select distinct B.CELLID

from tbATUDataNew as A ,tbATUDataNew as B

where A.LONGITUDE = 111.82841 and A.CELLID='253903-0' and (B.LATITUDE between A.LATITUDE-0.001 and A.LATITUDE+0.001)



说明：此执行计划发现使用函数的查询开销更小一些，【索引应该不起作用】

之后我们删除了longitude的索引，因为需要观察的是函数对索引的影响，所以只留下了latitude的索引。发现结果依旧，不过出现了缺少索引的提示。

按照此提示重新加了索引

create index atu\_ind on tbATUDataNew(LONGITUDE,CELLID) include(LATITUDE)

select distinct B.CELLID from tbATUDataNew as A,tbATUDataNew as B

where A.LONGITUDE = 111.82841 and A.CELLID = '253903-0' and

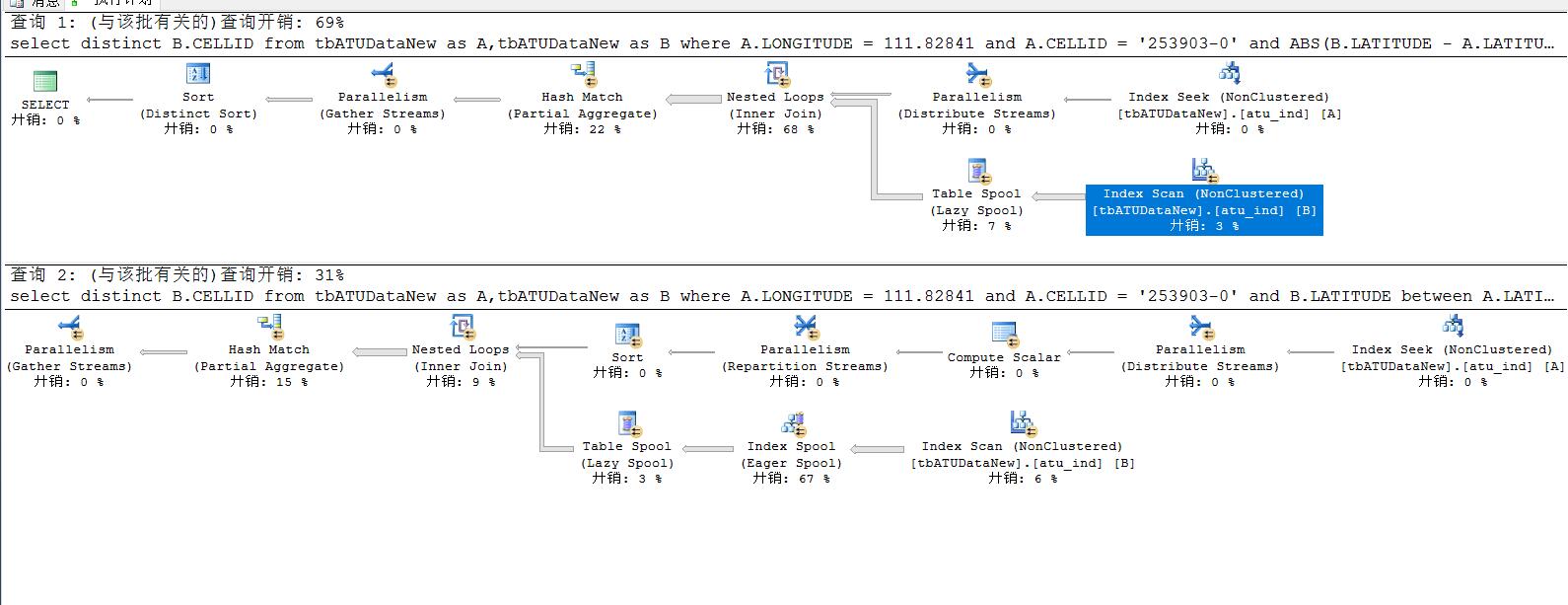
ABS(B.LATITUDE - A.LATITUDE) <= 0.001

select distinct B.CELLID from tbATUDataNew as A,tbATUDataNew as B

where A.LONGITUDE = 111.82841 and A.CELLID = '253903-0' and

B.LATITUDE between A.LATITUDE-0.001 and A.LATITUDE+0.001

查看执行计划



发现此时有函数的性能更低，但仍用到了索引查询。

### 8.3.5 多表嵌入式SQL查询

##### 实验要求

对访问相同的两张表且查询需求完全一样的2条select语句，一条使用嵌套查询，一条使用连接查询，实现两个要求之一：

1. 根据优化小区/保护带小区表和小区PCI优化调整结果表，查询小区类型为“优化区”的小区经调整后的PCI。

连接：

select PCI

from tbPCIAssignment join tbOptCell on tbOptCell.SECTOR\_ID=tbPCIAssignment.SECTOR\_ID

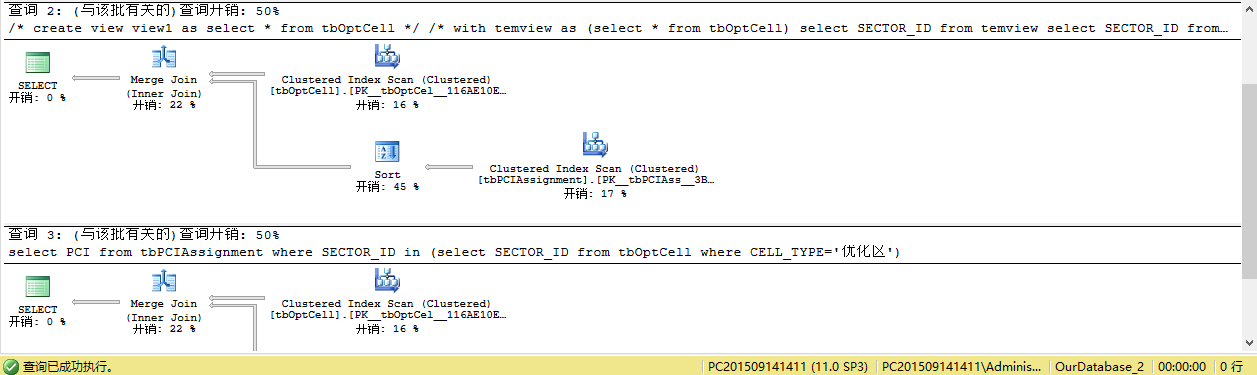
where CELL\_TYPE='优化区'

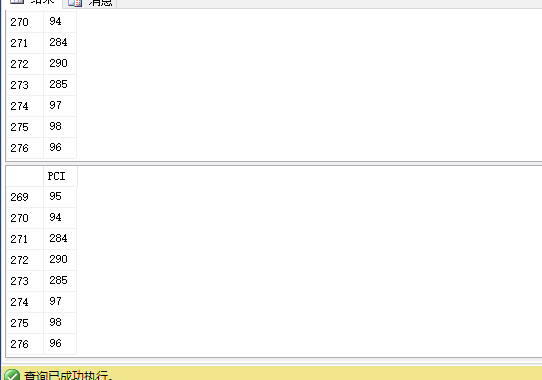
嵌套：

select PCI

from tbPCIAssignment

where SECTOR\_ID in (select SECTOR\_ID from tbOptCell where CELL\_TYPE='优化区')





说明：通过执行计划发现两种性能相同。不过一般在使用多表查询的时候应该避免使用嵌套查询，不过在实际应用中会发现性能相同甚至嵌套更优。

### 8.3.6 where查询条件中复合查询条件OR对索引的影响

##### 实验要求

在做这部分实验时，要求使用tbATUData表,按照下述步骤完成实验内容。在tbATUDatanew上创建一个索引，查询条件中为“A OR B”，A为创建的索引条件，B为非索引条件，提交select语句并查看执行计划中是否还能用到索引。再将含有“A OR B”查询条件的语句改为等价的两条select语句的union，对比这两种实现方式的执行效率。

##### 实验步骤

1. 创建tbATUData的备份表tbATUDatanew；
2. 在tbATUDatanew上创建索引；

create index stu\_ind2 on tbATUDataNew(Longitude)

select \*

from tbATUDataNew

where LONGITUDE = 111.87039 or LATITUDE = 34.74746

select \*

from tbATUDataNew

where LONGITUDE = 111.87039

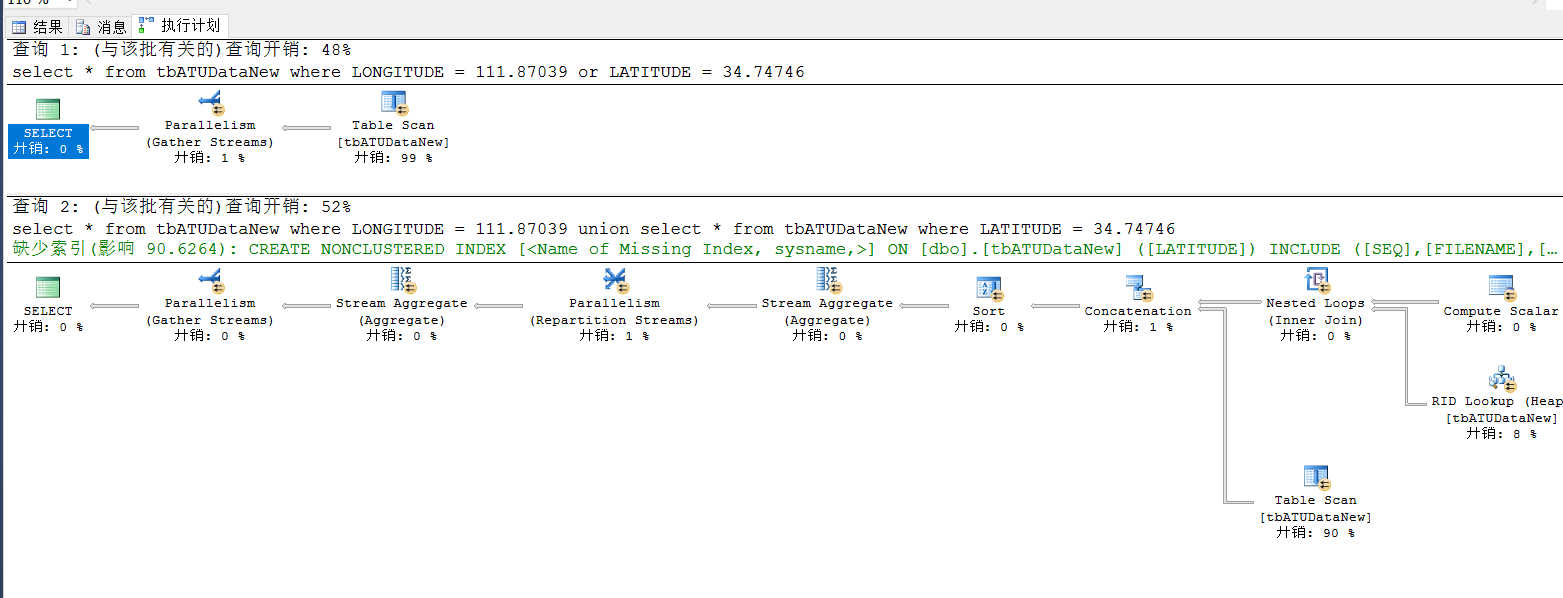
union

select \*

from tbATUDataNew

where LATITUDE = 34.74746

1. 编写select语句，where查询条件为“A OR B”，A为定义在索引属性上的查询条件，B为定义在非索引属性上的查询条件；将语句提交给DBMS执行后，查看执行计划是否使用了索引；



longitude上有索引，latitude上没有索引

说明：通过执行计划发现含有union的语句执行效率更低。是因为后一个查询步骤繁多，虽然使用索引，但是却还是在where latitude=34.74746时进行了全表扫描，效率降低。

### 8.3.7 聚集运算中的索引设计

##### 实验要求

选定作为聚集运算查询对象的关系表，对group by操作的分组属性，建立聚集/非聚集索引，对聚集运算（如count、sum、avg）的属性，建立非聚集索引。分析比较等价的有索引聚集运算、无索引聚集运算查询在查询执行计划、执行速度方面的区别。完成以下两个要求之一：

##### 实验步骤

在路测ATU数据表tbATUData，计算同一纬度的小区的平均经度值是多少并按纬度分组（注意索引的建立）；

1. 创建备份表；
2. 根据查询要求在备份表上创建索引；

create index lat on tbATUDataNew(LATITUDE)

create index long on tbATUDataNew(LONGITUDE)

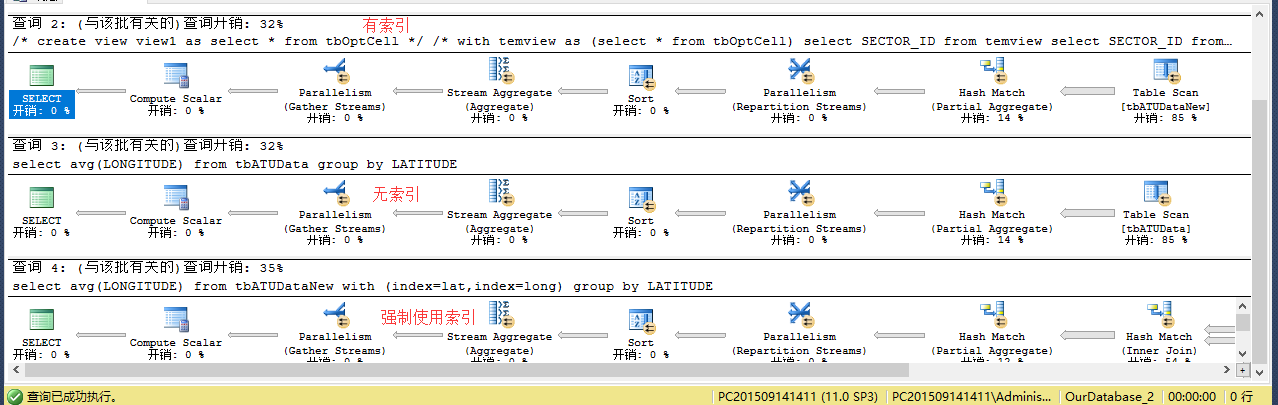
1. 根据查询要求，在原表和备份表上编写相同的select语句并将它们同时提交给DBMS；

select avg(LONGITUDE) from tbATUDataNew group by LATITUDE

select avg(LONGITUDE) from tbATUData group by LATITUDE

select avg(LONGITUDE) from tbATUDataNew with (index=lat,index=long) group by LATITUDE

1. 查询执行计划，分析比较等价的有索引聚集运算、无索引聚集运算查询在查询执行计划、执行速度方面的区别。



说明：强制使用索引的性能最差，有索引和无索引的性能相当。

再尝试建立聚集索引：

create clustered index atu\_index\_cluster on tbATUDataNew(LATITUDE)

create index atu\_index7\_noncluster on tbATUDataNew(LONGITUDE)

有索引

select LATITUDE,avg(LONGITUDE) as avg\_LONGITUDE

from tbATUDataNew

group by LATITUDE

强制使用索引

select LATITUDE,avg(LONGITUDE) as avg\_LONGITUDE

from tbATUDataNew with (index = atu\_index\_cluster,index = atu\_index7\_noncluster)

group by LATITUDE

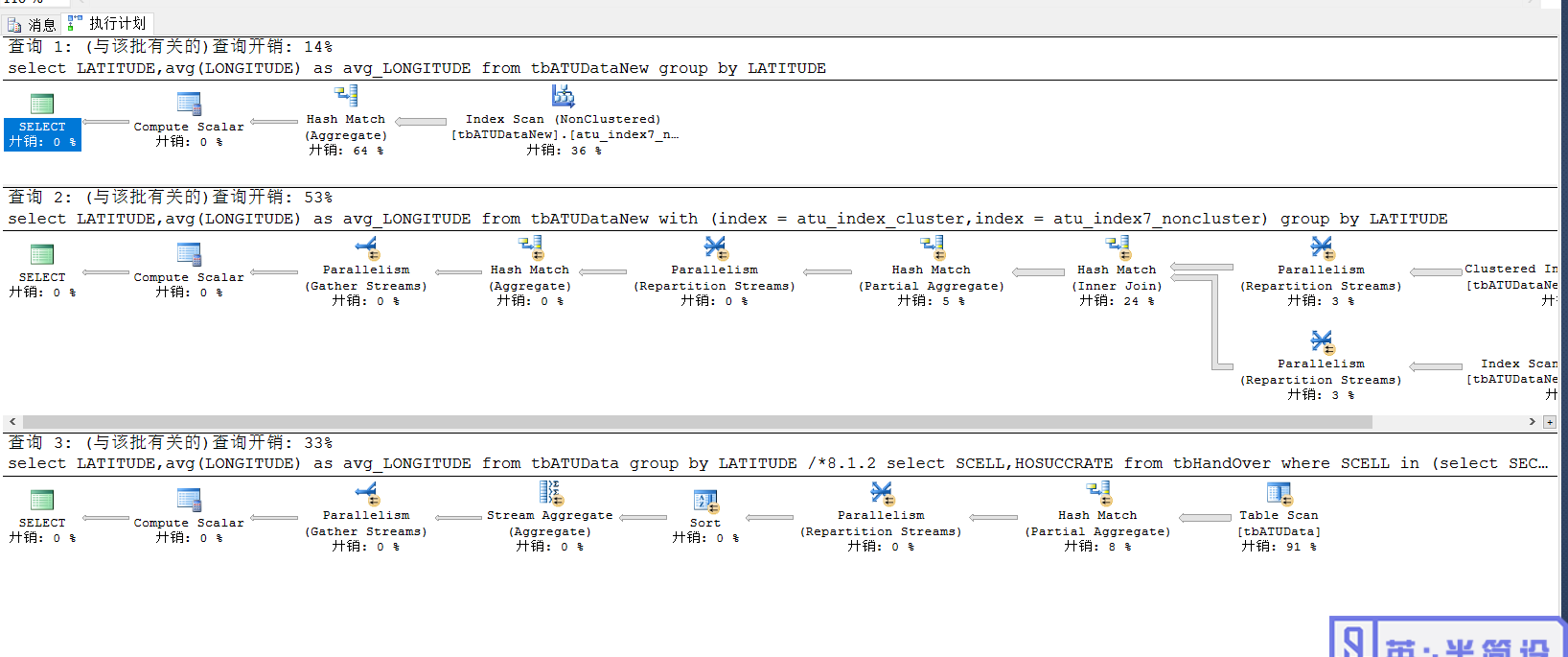
无索引

select LATITUDE,avg(LONGITUDE) as avg\_LONGITUDE

from tbATUData

group by LATITUDE

查看执行计划



说明：虽然有索引的表在查询时没有使用索引，但仍然比无索引的表查询快，在有索引的表上强制使用索引后，查询速度为三者中最慢。查询速度：有索引的表>无索引的表>强制使用索引的表。

### 8.3.8 Select子句中有无distinct的区别

##### 实验要求

在做这部分实验时，按照下述步骤完成实验内容。对访问相同的两张表且查询需求完全一样的2条select语句，一条使用distinct，一条不使用distinct，实现两个要求之一：

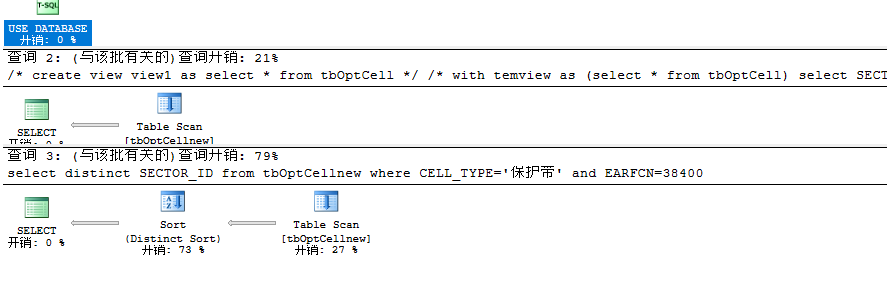
1. 根据优化小区/保护带小区表，查询频点编号为38400且小区类型为保护带的小区ID。

select SECTOR\_ID from tbOptCellnew

where CELL\_TYPE='保护带' and EARFCN=38400

select distinct SECTOR\_ID from tbOptCellnew

where CELL\_TYPE='保护带' and EARFCN=38400



说明：观察执行计划，发现distinct还要进行一遍sort，所以开销会大。

### 8.3.9 union、union all的区别

##### 实验要求

在做这部分实验时，按照下述步骤完成实验内容。对访问相同的两张表且查询需求完全一样的2条select语句，一条使用union，一条使用union，实现两个要求之一：

1. 找出所有二阶（同频）邻区关系表，一阶邻区关系表邻小区ID。

select N\_SECTOR\_ID from tbAdjCellnew

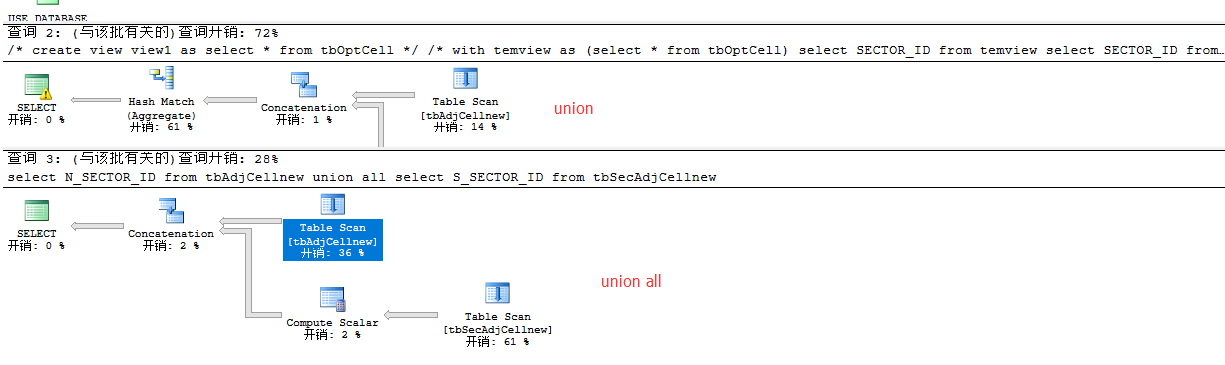
union

select S\_SECTOR\_ID from tbSecAdjCellnew

select N\_SECTOR\_ID from tbAdjCellnew

union all

select S\_SECTOR\_ID from tbSecAdjCellnew



说明：union因为要进行重复值扫描，所以效率会比union all低。

### 8.3.10 from中存在多余的关系表，即查询非最简化

##### 实验要求

在做这部分实验时，按照下述步骤完成实验内容。对访问相同的两张表且查询需求完全一样的2条等价select语句，一条在from后面增加一个多余的关系表（但在where子句中增加连接条件），另一条from后面最简化。

可自行选择查询语句，要求至少两表连接查询。分别判断2条语句的执行效果和执行速度，解释执行时间出现差异的原因。

没有多余关系表

select distinct tbAdjCellnew.N\_SECTOR\_ID

from tbAdjCellnew,tbSecAdjCellnew

where tbAdjCellnew.S\_SECTOR\_ID=tbSecAdjCellnew.S\_SECTOR\_ID

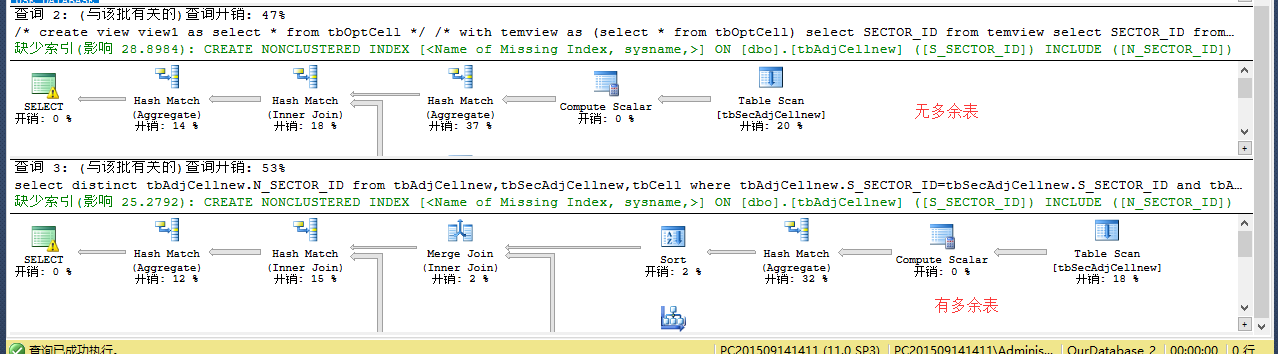
增加了一个多余的关系表tbCell和对应的连接条件

select distinct tbAdjCellnew.N\_SECTOR\_ID

from tbAdjCellnew,tbSecAdjCellnew,tbCell

where tbAdjCellnew.S\_SECTOR\_ID=tbSecAdjCellnew.S\_SECTOR\_ID and tbAdjCellnew.S\_SECTOR\_ID=tbCell.SECTOR\_ID

查看执行计划



我们还实验了另外一个表

没有多余关系表

select distinct EARFCN

from tbOptCell,tbHandOver

where tbOptCell.SECTOR\_ID = tbHandOver.SCELL

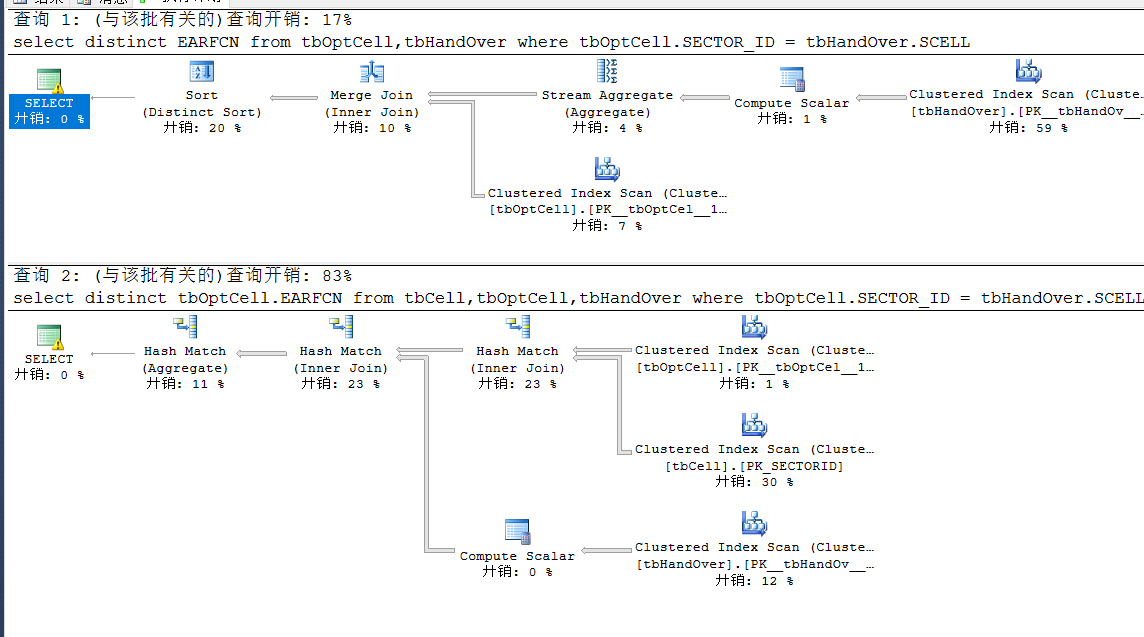
增加了一个多余的关系表tbATUData和对应的连接条件

select distinct tbOptCell.EARFCN

from tbCell,tbOptCell,tbHandOver

where tbOptCell.SECTOR\_ID = tbHandOver.SCELL and tbCell.SECTOR\_ID = tbOptCell.SECTOR\_ID

查看执行计划



说明：增加多余表会增大开销，而增大开销的程度与选择的表有关。数据量较大的表增大开销的程度较大。

### 实验总结

在实验中有哪些重要问题或者事件？你如何处理的？你的收获是什么？有何建议和意见等等。

本次实验中，我们遇到了以下问题：

1. 在8.3.1测试非最左索引有没有作用时，发现使用了索引的性能还是优于未使用索引的性能。之后测试了第三索引，发现索引的确没有起到作用，猜测第二索引时可能和数据有关系。具体解释见以上实验报告该环节。
2. 在8.3.4测试使用函数和不使用函数对索引和性能的影响时，我们发现使用函数虽然使索引失效了，但其性能反而更高一些。不使用函数时查询RID花费和开销很大，之后去掉这一环节后，发现的确使用函数会使索引失效，性能较差。
3. 在8.3.7 测试有无索引和强制索引的聚集运算性能比较时，当都加非聚集索引时，发现有无索引的聚集运算性能相同。之后加了聚集索引，得到了有索引的性能优于无索引的结果。

总结：在本次实验中，我们通过测试比较对于查询优化有了进一步的认识。通过观察执行计划，我们也大概理解了开销主要花费在哪里，通过优化，降低这部分开销，提升性能。我们通过遇到的问题也了解到所谓优化，在对于不同的数据量或是选择的不同数据时是有差别的，并不是永远都能起到优化的作用。不过在此次实验中，还是能看到有些语句的选择是会对性能有很大影响的，尤其对于数据量大的表，在之后编写SQL语句中也会注意到这一点，尽可能将语句优化，降低开销。