《数据查询与修改》
实验报告

学院：	计算机学院（国家示范性软件学院）
班级：	2019211308 2019211308 2019211308
姓名：	顾天阳 曾世茂 庞仕泽
学号：	2019211539 2019211532 2019211509

目录

一、实验环境

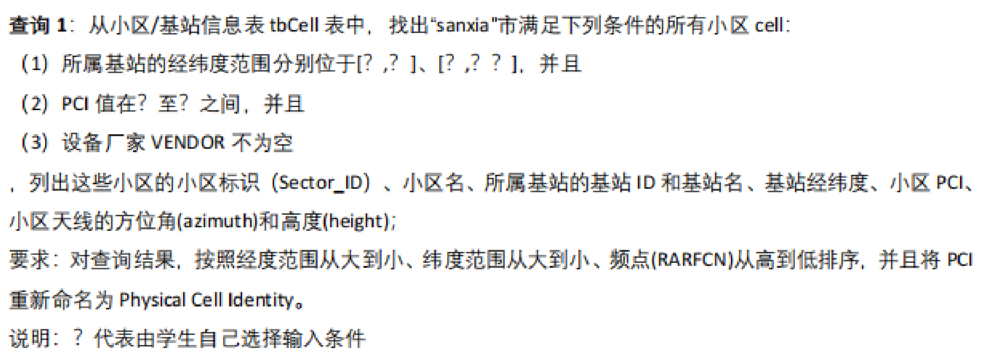
本次 GaussDB 数据查询与修改实验基于 GaussDB(for openGauss) 1.4.10 进行，本地操作系统环境为 Windows 11 Home。

二、实验内容

1、 单表查询

● 查询 1

(1) 查询内容



查询 1：从小区/基站信息表 tbCell 表中，找出“sanxia”市满足下列条件的所有小区 cell：

（1）所属基站的经纬度范围分别位于[? ,?]、[? ,? ?]，并且

（2）PCI 值在? 至? 之间，并且

（3）设备厂家 VENDOR 不为空

，列出这些小区的小区标识（Sector_ID）、小区名、所属基站的基站 ID 和基站名、基站经纬度、小区 PCI、小区天线的方位角(azimuth)和高度(height)；

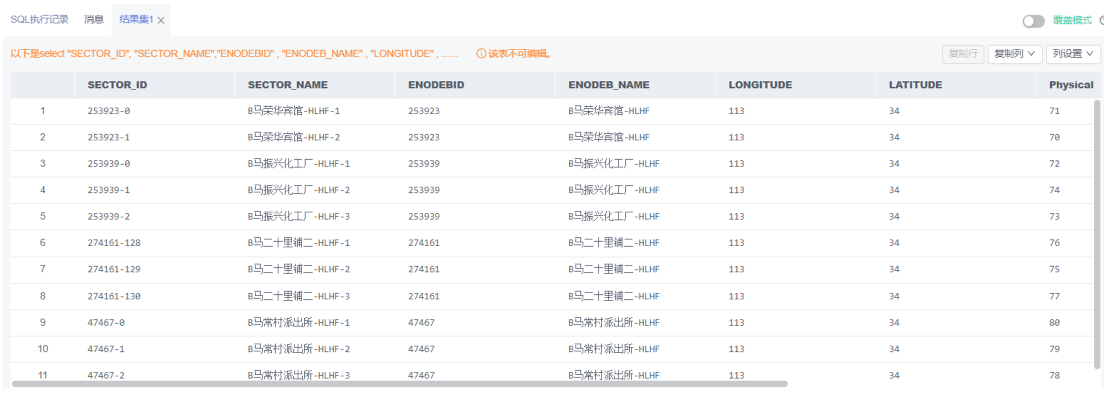
要求：对查询结果，按照经度范围从大到小、纬度范围从大到小、频点(RARFCN)从高到低排序，并且将 PCI 重新命名为 Physical Cell Identity。

说明：？代表由学生自己选择输入条件

(2) 查询语句

```sql
select "SECTOR_ID", "SECTOR_NAME","ENODEBID" , "ENODEB_NAME" , "LONGITUDE" , "LATITUDE","PCI" AS "Physical Cell Identity","AZIMUTH","HEIGHT" from tbcell where
"CITY" = 'sanxia' AND
"LONGITUDE" >= 112 AND
"LONGITUDE" <= 114 AND
"PCI" >= 70 AND
"PCI" <= 80 AND
"LATITUDE" >= 34 AND
"LATITUDE" <= 35 AND
"VENDOR" NOTNULL
ORDER BY "LONGITUDE","LATITUDE","EARFCN" DESC
```

(3) 查询结果


SQL执行记录　消息　结果集1

以下是select "SECTOR_ID", "SECTOR_NAME","ENODEBID" , "ENODEB_NAME" , "LONGITUDE" , ...　ⓘ 该表不可编辑。

	SECTOR_ID	SECTOR_NAME	ENODEBID	ENODEB_NAME	LONGITUDE	LATITUDE	Physical
1	253923-0	B马荣华宾馆-HLHF-1	253923	B马荣华宾馆-HLHF	113	34	71
2	253923-1	B马荣华宾馆-HLHF-2	253923	B马荣华宾馆-HLHF	113	34	70
3	253939-0	B马振兴化工厂-HLHF-1	253939	B马振兴化工厂-HLHF	113	34	72
4	253939-1	B马振兴化工厂-HLHF-2	253939	B马振兴化工厂-HLHF	113	34	74
5	253939-2	B马振兴化工厂-HLHF-3	253939	B马振兴化工厂-HLHF	113	34	73
6	274161-128	B马二十里铺二-HLHF-1	274161	B马二十里铺二-HLHF	113	34	76
7	274161-129	B马二十里铺二-HLHF-2	274161	B马二十里铺二-HLHF	113	34	75
8	274161-130	B马二十里铺二-HLHF-3	274161	B马二十里铺二-HLHF	113	34	77
9	47467-0	B马常村派出所-HLHF-1	47467	B马常村派出所-HLHF	113	34	80
10	47467-1	B马常村派出所-HLHF-2	47467	B马常村派出所-HLHF	113	34	79
11	47467-2	B马常村派出所-HLHF-3	47467	B马常村派出所-HLHF	113	34	78

- 查询 2

（1）查询内容

查询 2：从小区/基站信息表 tbCell 表中，找出“sanxia”市满足下列条件的所有基站 ENodeB：

（1）所属基站的经纬度范围分别位于[? -?]、[? ,? ?]，

（2）属于该基站的小区中，至少有一个小区的 PCI 值在？至？之间

，列出这些基站的基站 ID 和基站名、基站经纬度、基站类型(Style)、设备生产厂家(Vendor)；要求：对查询结果，按照基站位置从北到南、从东到西排序，并且对查询结果使用 distinct 去重。

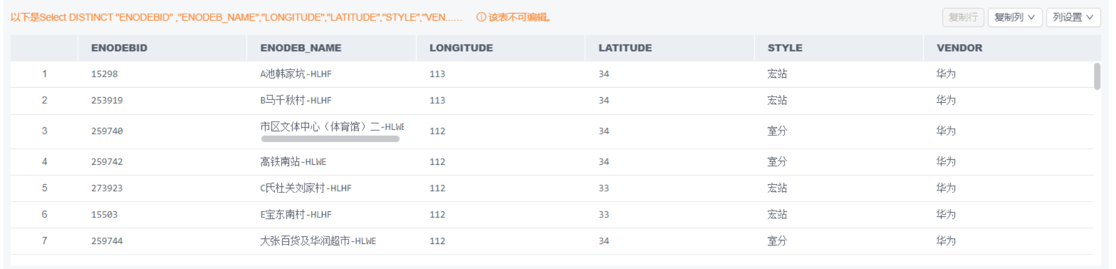
比较对查询结果去重和不去重，在查询时间和查询结果上的差异。

（2）查询

① 去重

```sql
Select DISTINCT  "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VENDOR"
From(select *
From tbcell_2
Where "PCI" between 40 and 50)
```

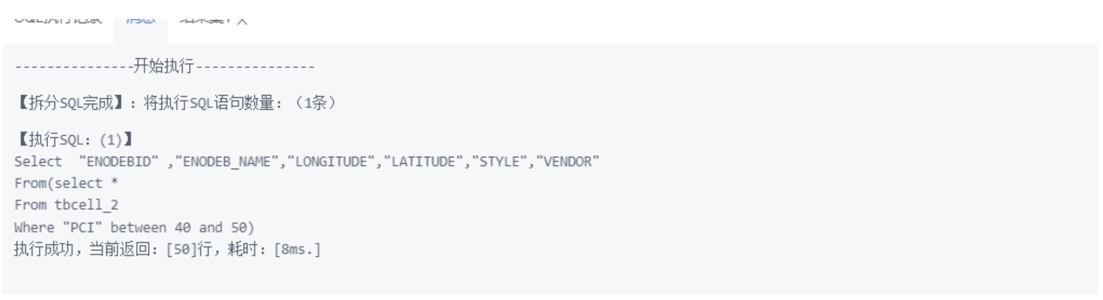
```
--------------开始执行--------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
Select DISTINCT  "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VENDOR"
From(select *
From tbcell_2
Where "PCI" between 40 and 50)
执行成功，当前返回：[50]行，耗时：[24ms.]
```


以下是Select DISTINCT "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VEN......　ⓘ 该表不可编辑。

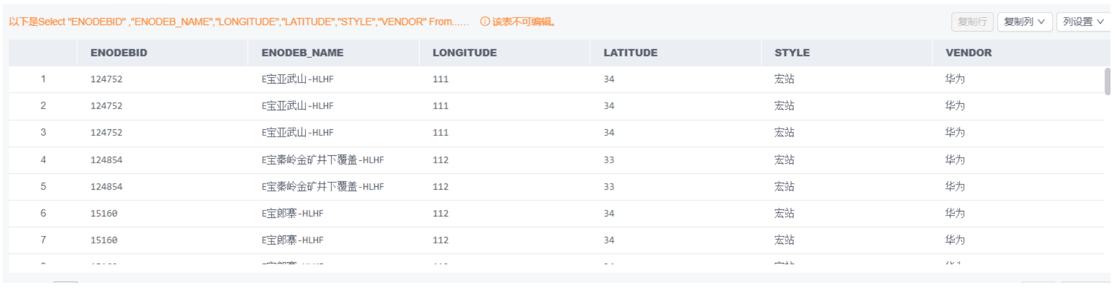
	ENODEBID	ENODEB_NAME	LONGITUDE	LATITUDE	STYLE	VENDOR
1	15298	A池韩家坑-HLHF	113	34	宏站	华为
2	253919	B马千秋村-HLHF	113	34	宏站	华为
3	259740	市区文体中心（体育馆）二-HLWE	112	34	室分	华为
4	259742	高铁南站-HLWE	112	34	室分	华为
5	273923	C氏杜关刘家村-HLHF	112	33	宏站	华为
6	15503	E宝东南村-HLHF	112	33	宏站	华为
7	259744	大张百货及华润超市-HLWE	112	34	室分	华为

② 不去重

```sql
Select  "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VENDOR"
From(select *
From tbcell_2
Where "PCI" between 40 and 50)
```



```
--------------开始执行--------------
【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】
Select  "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VENDOR"
From(select *
From tbcell_2
Where "PCI" between 40 and 50)
执行成功, 当前返回: [50]行, 耗时: [8ms.]
```



以下是Select "ENODEBID" ,"ENODEB_NAME","LONGITUDE","LATITUDE","STYLE","VENDOR" From...... ⓘ 该表不可编辑。

	ENODEBID	ENODEB_NAME	LONGITUDE	LATITUDE	STYLE	VENDOR
1	124752	E宝亚武山-HLHF	111	34	宏站	华为
2	124752	E宝亚武山-HLHF	111	34	宏站	华为
3	124752	E宝亚武山-HLHF	111	34	宏站	华为
4	124854	E宝秦岭金矿井下覆盖-HLHF	112	33	宏站	华为
5	124854	E宝秦岭金矿井下覆盖-HLHF	112	33	宏站	华为
6	15160	E宝郎寨-HLHF	112	34	宏站	华为
7	15160	E宝郎寨-HLHF	112	34	宏站	华为

结果：添加去重后消耗时间会大幅增加。

- 查询 3

(1) 查询内容

查询 3：从小区/基站信息表 tbCell 表中，找出满足下列条件的小区：

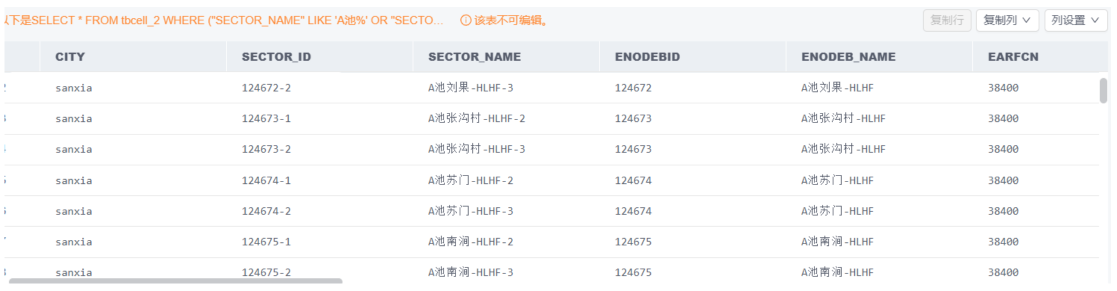
(1) 小区名开头部分包含"A 池"或"高铁"，或者基站名中包含"医院"或"实验高中"，并且

(2) 不是所属基站的第 1 小区，即小区名结尾部分不是"-1"

(2) 查询语句

```sql
SELECT *
FROM tbcell_2   WHERE
("SECTOR_NAME" LIKE 'A 池%'   OR
"SECTOR_NAME" LIKE '高铁%' OR
"ENODEB_NAME" LIKE '%医院%' OR
"ENODEB_NAME" LIKE '%实验高中%') AND
"SECTOR_NAME" NOT LIKE '%-1'
```

(3) 查询结果


以下是SELECT * FROM tbcell_2 WHERE ("SECTOR_NAME" LIKE 'A池%' OR "SECTO...   ⓘ 该表不可编辑。

	CITY	SECTOR_ID	SECTOR_NAME	ENODEBID	ENODEB_NAME	EARFCN
	sanxia	124672-2	A池刘果-HLHF-3	124672	A池刘果-HLHF	38400
	sanxia	124673-1	A池张沟村-HLHF-2	124673	A池张沟村-HLHF	38400
	sanxia	124673-2	A池张沟村-HLHF-3	124673	A池张沟村-HLHF	38400
	sanxia	124674-1	A池苏门-HLHF-2	124674	A池苏门-HLHF	38400
	sanxia	124674-2	A池苏门-HLHF-3	124674	A池苏门-HLHF	38400
	sanxia	124675-1	A池南涧-HLHF-2	124675	A池南涧-HLHF	38400
	sanxia	124675-2	A池南涧-HLHF-3	124675	A池南涧-HLHF	38400

- ▪查询 4

  (1)　查询内容

查询 4：从小区/基站信息表 tbCell 表中，找出满足下列条件的小区：

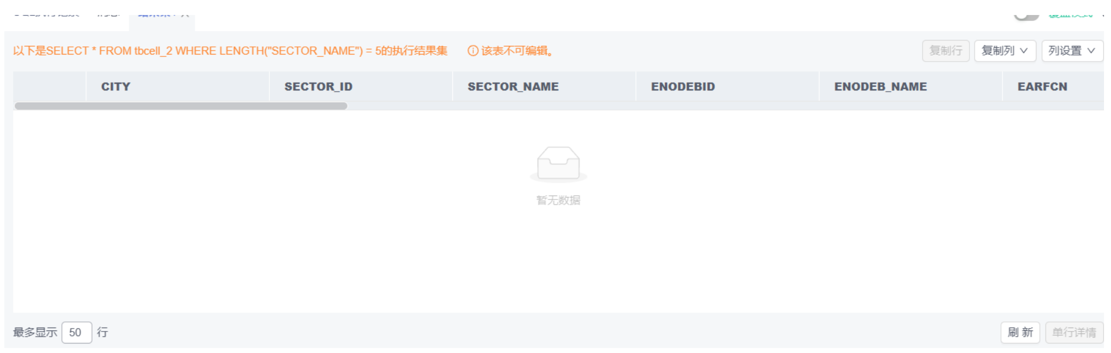
(1) 小区标识由 5 个字符组成，并且

(2) 小区所属基站的名字/标识至少包括 8 个字符，即名字字符串的长度不小于 8。

  (2)　查询语句

```sql
SELECT * FROM tbcell_2 WHERE
LENGTH("SECTOR_ID") = 5 AND
LENGTH("ENODEB_NAME") >= 8
```

  (3)　查询结果


以下是SELECT * FROM tbcell_2 WHERE LENGTH("SECTOR_NAME") = 5的执行结果集   ⓘ 该表不可编辑。

	CITY	SECTOR_ID	SECTOR_NAME	ENODEBID	ENODEB_NAME	EARFCN

暂无数据

最多显示 50 行

没有一条数据符合该要求

- 查询 5

  (1) 查询内容

查询 5：使用集合并操作 union、union all，从小区 KPI 指标表 tbCellKPI 查询满足下列条件的小区

(1)小区 RRC 建立成功率 qf (%)大于 95%，或者

(2)E-RAB 建立成功率 2 (%)大于 99%

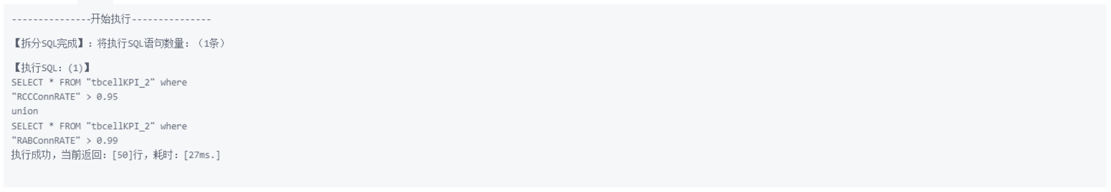
对比 union all、union 操作在查询结果、执行时间上的差异。

  (2) 查询

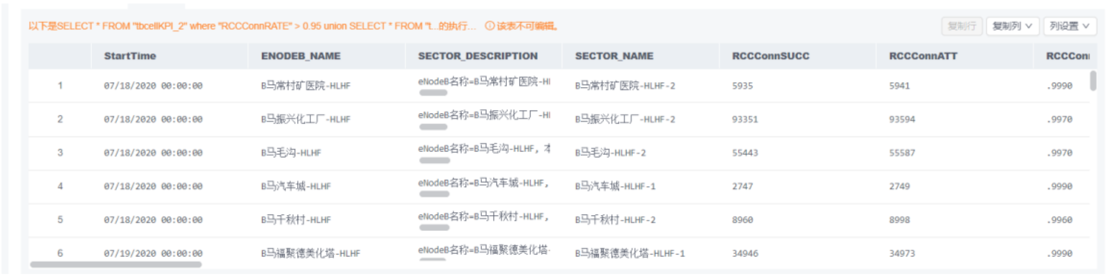
  ① Union

```sql
SELECT * FROM "tbcellKPI_2" where
```

```sql
"RCCConnRATE" > 0.95
union
SELECT * FROM "tbcellKPI_2" where
"RABConnRATE" > 0.99
```



```
--------------开始执行--------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT * FROM "tbcellKPI_2" where
"RCCConnRATE" > 0.95
union
SELECT * FROM "tbcellKPI_2" where
"RABConnRATE" > 0.99
执行成功，当前返回：[50]行，耗时：[27ms.]
```

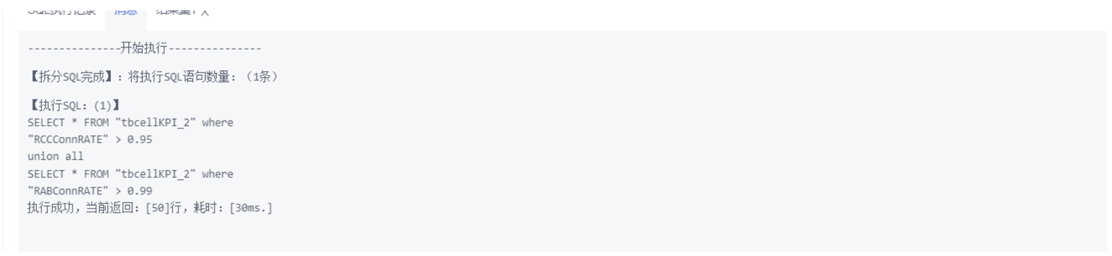


以下是SELECT * FROM "tbcellKPI_2" where "RCCConnRATE" > 0.95 union SELECT * FROM "t...的执行... ⓘ该表不可编辑。

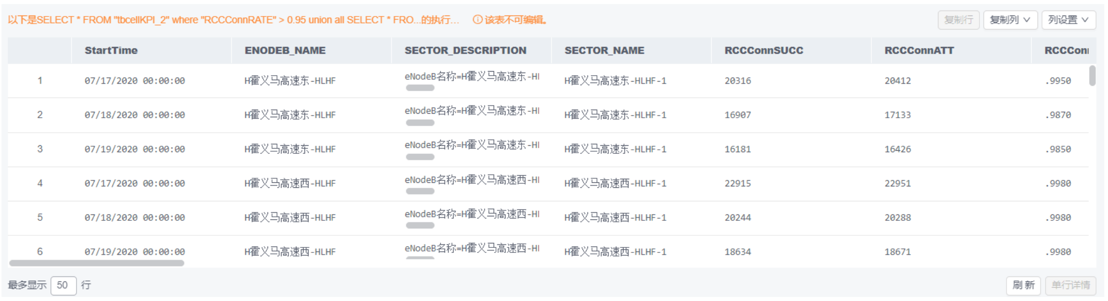
	StartTime	ENODEB_NAME	SECTOR_DESCRIPTION	SECTOR_NAME	RCCConnSUCC	RCCConnATT	RCCCon
1	07/18/2020 00:00:00	B马常村矿医院-HLHF	eNodeB名称=B马常村矿医院-HI	B马常村矿医院-HLHF-2	5935	5941	.9990
2	07/18/2020 00:00:00	B马振兴化工厂-HLHF	eNodeB名称=B马振兴化工厂-HI	B马振兴化工厂-HLHF-2	93351	93594	.9970
3	07/18/2020 00:00:00	B马毛沟-HLHF	eNodeB名称=B马毛沟-HLHF，才	B马毛沟-HLHF-2	55443	55587	.9970
4	07/18/2020 00:00:00	B马汽车城-HLHF	eNodeB名称=B马汽车城-HLHF，	B马汽车城-HLHF-1	2747	2749	.9990
5	07/18/2020 00:00:00	B马千秋村-HLHF	eNodeB名称=B马千秋村-HLHF，	B马千秋村-HLHF-2	8960	8998	.9960
6	07/19/2020 00:00:00	B马福聚德美化塔-HLHF	eNodeB名称=B马福聚德美化塔·	B马福聚德美化塔-HLHF-1	34946	34973	.9990

② Union all

```sql
SELECT * FROM "tbcellKPI_2" where
"RCCConnRATE" > 0.95
union all
SELECT * FROM "tbcellKPI_2" where
"RABConnRATE" > 0.99
```



```
--------------开始执行--------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT * FROM "tbcellKPI_2" where
"RCCConnRATE" > 0.95
union all
SELECT * FROM "tbcellKPI_2" where
"RABConnRATE" > 0.99
执行成功，当前返回：[50]行，耗时：[30ms.]
```



以下是SELECT * FROM "tbcellKPI_2" where "RCCConnRATE" > 0.95 union all SELECT * FRO...的执行... ⓘ该表不可编辑。

	StartTime	ENODEB_NAME	SECTOR_DESCRIPTION	SECTOR_NAME	RCCConnSUCC	RCCConnATT	RCCCon
1	07/17/2020 00:00:00	H霍义马高速东-HLHF	eNodeB名称=H霍义马高速东-HI	H霍义马高速东-HLHF-1	20316	20412	.9950
2	07/18/2020 00:00:00	H霍义马高速东-HLHF	eNodeB名称=H霍义马高速东-HI	H霍义马高速东-HLHF-1	16907	17133	.9870
3	07/19/2020 00:00:00	H霍义马高速东-HLHF	eNodeB名称=H霍义马高速东-HI	H霍义马高速东-HLHF-1	16181	16426	.9850
4	07/17/2020 00:00:00	H霍义马高速西-HLHF	eNodeB名称=H霍义马高速西-HI	H霍义马高速西-HLHF-1	22915	22951	.9980
5	07/18/2020 00:00:00	H霍义马高速西-HLHF	eNodeB名称=H霍义马高速西-HI	H霍义马高速西-HLHF-1	20244	20288	.9980
6	07/19/2020 00:00:00	H霍义马高速西-HLHF	eNodeB名称=H霍义马高速西-HI	H霍义马高速西-HLHF-1	18634	18671	.9980

最多显示 50 行

结果： Union 会执行去重操作，所以 Union all 中会出现重复的数据。而在这个例子中，两者的运行时间没有明显差异

- 查询 6

(1)　查询内容

查询 6：结合教材 3.4.1 节元组变量样例，使用集合操作 except、except all，从小区/基站信息表 tbCell 表中，查询位于最北端（具有最大纬度）的基站。

　　对比使用 except、except all、聚集函数 max，完成此查询在执行时间、查询结果上的异同。

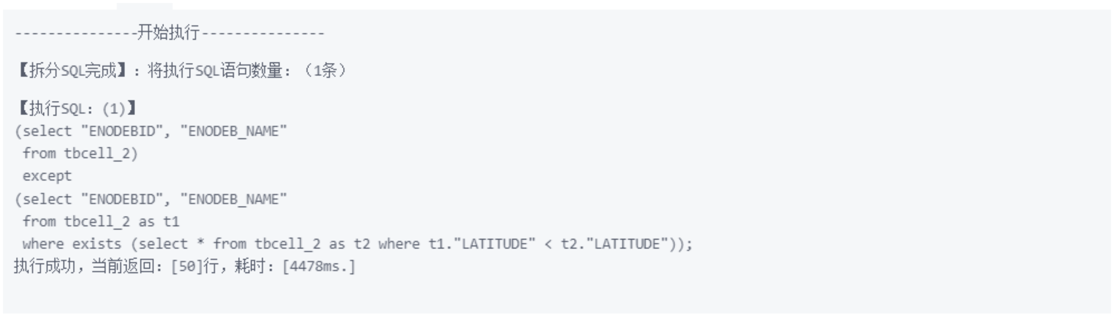
(2)　查询

①　Except

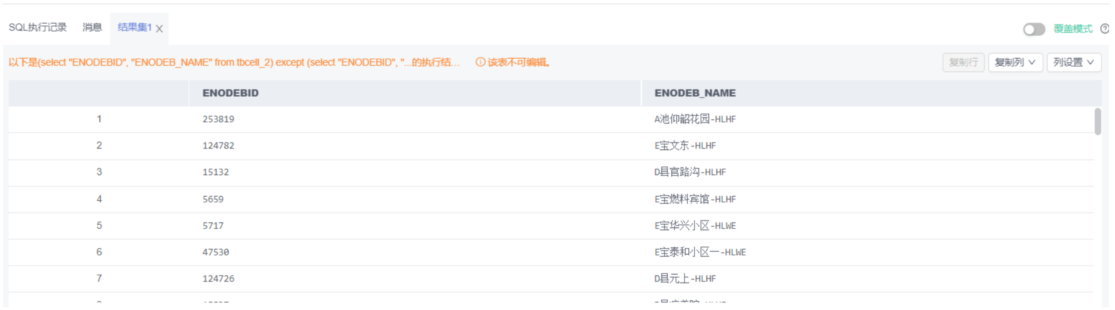
查询语句

```
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2)
 except
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2 as t1
where exists (select * from tbcell_2 as t2 where t1."LATITUDE" < t2."LATITUDE"));
```

查询结果



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2)
 except
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2 as t1
 where exists (select * from tbcell_2 as t2 where t1."LATITUDE" < t2."LATITUDE"));
执行成功，当前返回：[50]行，耗时：[4478ms.]
```



以下是(select "ENODEBID", "ENODEB_NAME" from tbcell_2) except (select "ENODEBID", "...的执行结... ⓘ 该表不可编辑。

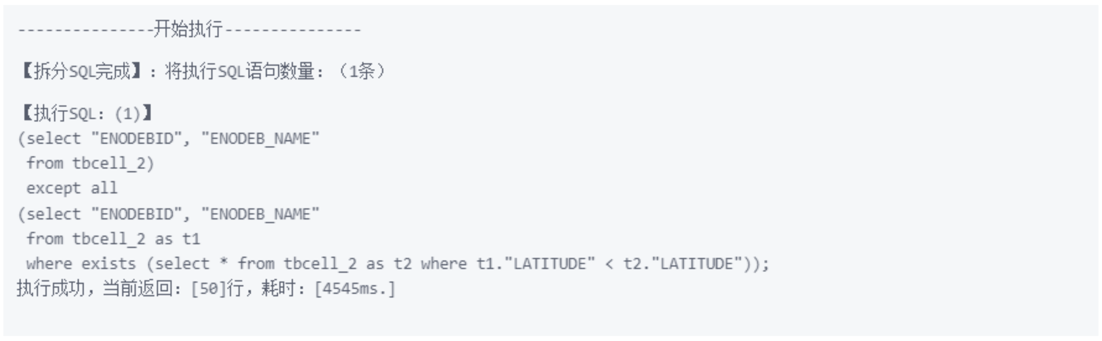
	ENODEBID	ENODEB_NAME
1	253819	A池仰韶花园-HLHF
2	124782	E宝文东-HLHF
3	15132	D县官路沟-HLHF
4	5659	E宝燃料宾馆-HLHF
5	5717	E宝华兴小区-HLWE
6	47530	E宝泰和小区一-HLWE
7	124726	D县元上-HLHF

②　Except all

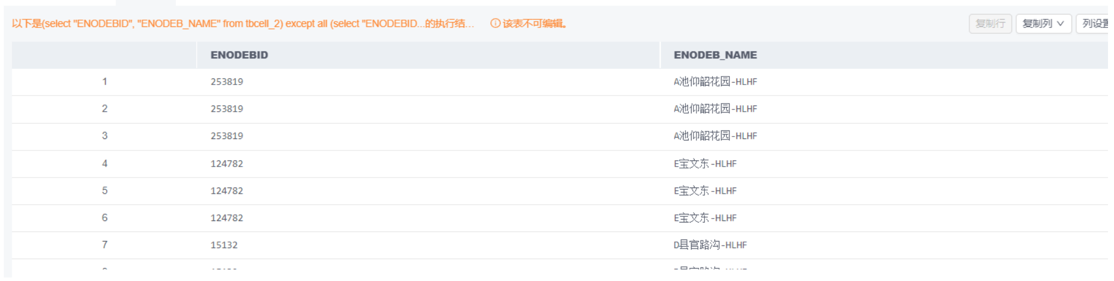
查询语句

```
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2)
 except all
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2 as t1
 where exists (select * from tbcell_2 as t2 where t1."LATITUDE" < t2."LATITUDE"));
```

查询结果



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2)
 except all
(select "ENODEBID", "ENODEB_NAME"
 from tbcell_2 as t1
 where exists (select * from tbcell_2 as t2 where t1."LATITUDE" < t2."LATITUDE"));
执行成功，当前返回：[50]行，耗时：[4545ms.]
```



以下是(select "ENODEBID", "ENODEB_NAME" from tbcell_2) except all (select "ENODEBID...的执行结... ⓘ该表不可编辑。

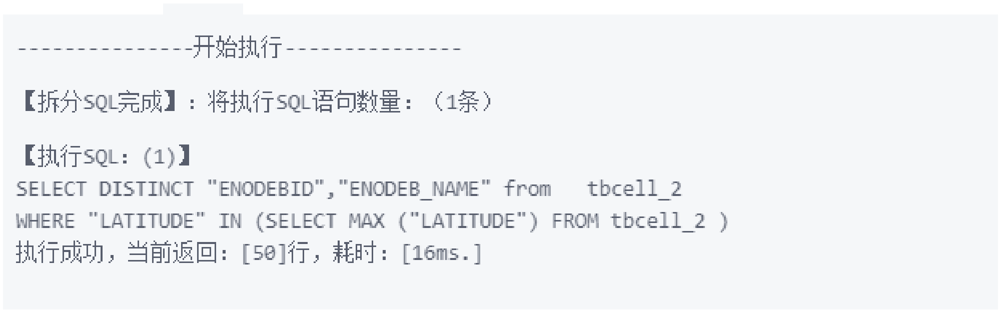
	ENODEBID	ENODEB_NAME
1	253819	A池仰韶花园-HLHF
2	253819	A池仰韶花园-HLHF
3	253819	A池仰韶花园-HLHF
4	124782	E宝文东-HLHF
5	124782	E宝文东-HLHF
6	124782	E宝文东-HLHF
7	15132	D县官路沟-HLHF

③ Max

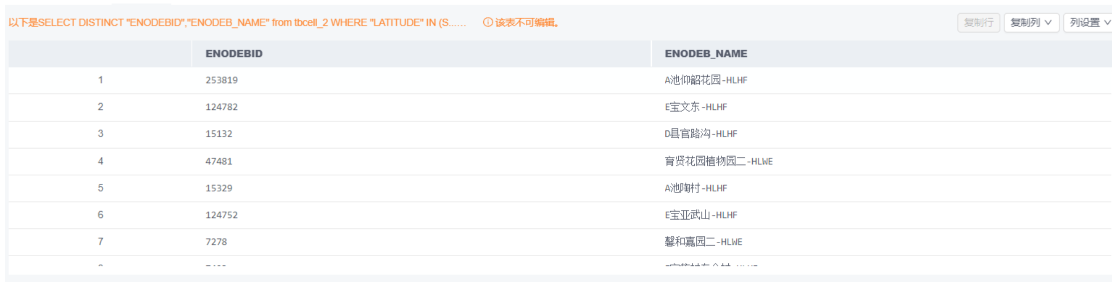
查询语句

```sql
SELECT distinct "ENODEBID", "ENODEB_NAME" FROM tbcell_2
WHERE "LATITUDE" IN(SELECT MAX("LATITUDE") FROM tbcell_2)
```

查询结果



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT DISTINCT "ENODEBID","ENODEB_NAME" from   tbcell_2
WHERE "LATITUDE" IN (SELECT MAX ("LATITUDE") FROM tbcell_2 )
执行成功，当前返回：[50]行，耗时：[16ms.]
```



以下是SELECT DISTINCT "ENODEBID","ENODEB_NAME" from tbcell_2 WHERE "LATITUDE" IN (S...... ⓘ该表不可编辑。

	ENODEBID	ENODEB_NAME
1	253819	A池仰韶花园-HLHF
2	124782	E宝文东-HLHF
3	15132	D县官路沟-HLHF
4	47481	育贤花园植物园二-HLWE
5	15329	A池陶村-HLHF
6	124752	E宝亚武山-HLHF
7	7278	馨和嘉园二-HLWE

结果分析：直接使用 max 函数的速度最快，远超使用 except 或 except all。而直接使用 max 函数且使用 distinct 后结果无重复值，使用 except all 可能存在重复值，except 也不会存在重复值

2、 多表查询

- 查询 7

(1) 查询内容

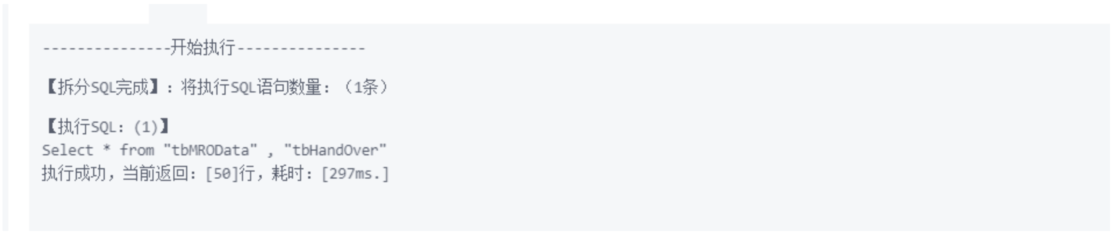
查询 7：选取两张数据量比较大的表 T_1 和 T_2，如 tbMROData、tbCellTraffic、tbC2I、tbHandover，执行如下无连接条件的笛卡尔积操作，观察数据库系统的反应和查询结果：

Select * from T_1, T_2

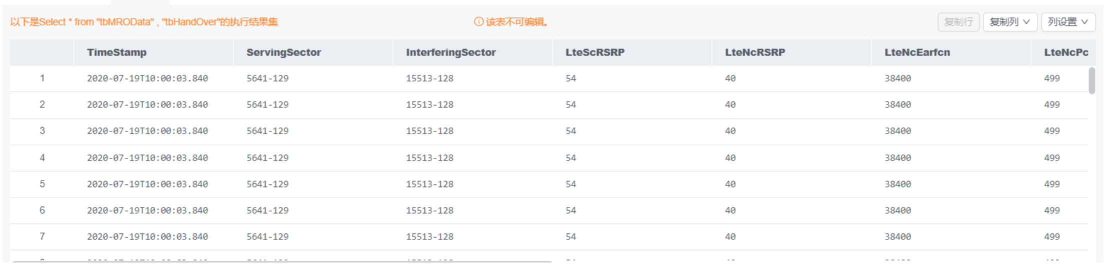
(2) 查询语句

```sql
Select * from "tbMROData" , "tbHandOver"
```

(3) 查询结果



```
--------------开始执行--------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
Select * from "tbMROData" , "tbHandOver"
执行成功，当前返回：[50]行，耗时：[297ms.]
```



以下是Select * from "tbMROData" , "tbHandOver"的执行结果集　　ⓘ 该表不可编辑。

	TimeStamp	ServingSector	InterferingSector	LteScRSRP	LteNcRSRP	LteNcEarfcn	LteNcPc
1	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
2	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
3	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
4	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
5	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
6	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499
7	2020-07-19T10:00:03.840	5641-129	15513-128	54	40	38400	499

结果分析：两表连接的速度较慢。 连接结果两个表的全连接

- 查询 8

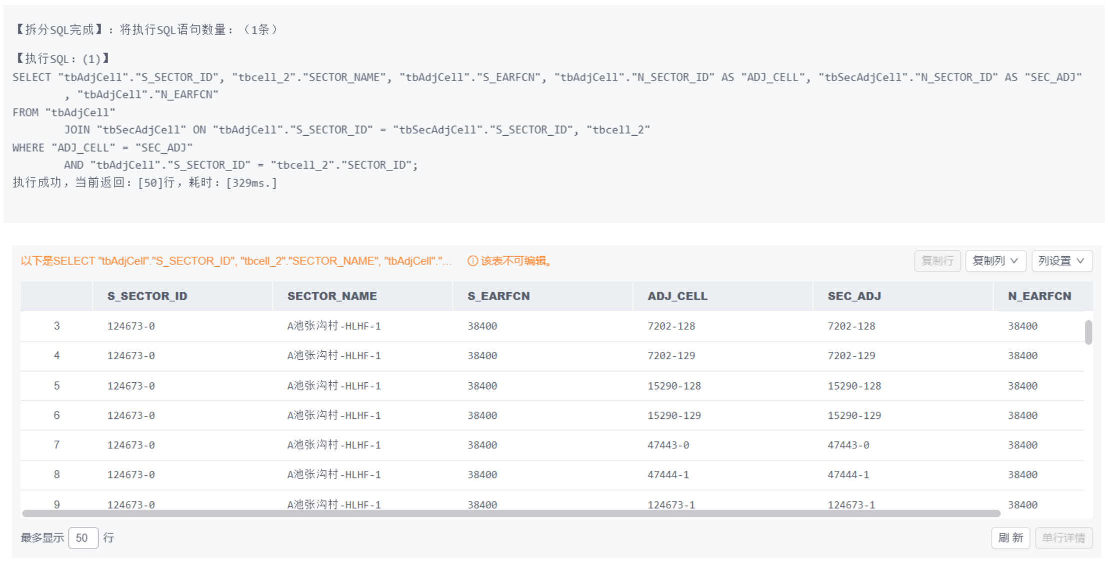
(1) 查询内容

查询 8：使用多表连接操作（3.3.3 join/natual join，4.1.1 join），从小区/基站信息表 tbCell 表、小区一阶邻区关系表 tbAdjCell、小区二阶（同频）邻区关系表 tbSecAdjCell 中，查询有相同的一阶邻小区和二阶邻小区的主小区，列出这些主小区的小区标识、小区名称、小区频点，以及该小区的一阶邻小区和二阶邻小区的小区标识及其频点。

(2) 查询语句

```sql
SELECT "tbAdjCell"."S_SECTOR_ID", "tbcell_2"."SECTOR_NAME", "tbAdjCell"."S_EARFCN", "tbAdjCell"."N_SECTOR_ID" AS "ADJ_CELL", "tbSecAdjCell"."N_SECTOR_ID" AS "SEC_ADJ"
    , "tbAdjCell"."N_EARFCN"
FROM "tbAdjCell"
    JOIN "tbSecAdjCell" ON "tbAdjCell"."S_SECTOR_ID" = "tbSecAdjCell"."S_SECTOR_ID", "tbcell_2"
WHERE "ADJ_CELL" = "SEC_ADJ"
    AND "tbAdjCell"."S_SECTOR_ID" = "tbcell_2"."SECTOR_ID";
```

(3) 查询结果



```
【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT "tbAdjCell"."S_SECTOR_ID", "tbcell_2"."SECTOR_NAME", "tbAdjCell"."S_EARFCN", "tbAdjCell"."N_SECTOR_ID" AS "ADJ_CELL", "tbSecAdjCell"."N_SECTOR_ID" AS "SEC_ADJ"
        , "tbAdjCell"."N_EARFCN"
FROM "tbAdjCell"
        JOIN "tbSecAdjCell" ON "tbAdjCell"."S_SECTOR_ID" = "tbSecAdjCell"."S_SECTOR_ID", "tbcell_2"
WHERE "ADJ_CELL" = "SEC_ADJ"
        AND "tbAdjCell"."S_SECTOR_ID" = "tbcell_2"."SECTOR_ID";
执行成功，当前返回：[50]行，耗时：[329ms.]
```

以下是SELECT "tbAdjCell"."S_SECTOR_ID", "tbcell_2"."SECTOR_NAME", "tbAdjCell"."... ⓘ 该表不可编辑。

	S_SECTOR_ID	SECTOR_NAME	S_EARFCN	ADJ_CELL	SEC_ADJ	N_EARFCN
3	124673-0	A池张沟村-HLHF-1	38400	7202-128	7202-128	38400
4	124673-0	A池张沟村-HLHF-1	38400	7202-129	7202-129	38400
5	124673-0	A池张沟村-HLHF-1	38400	15290-128	15290-128	38400
6	124673-0	A池张沟村-HLHF-1	38400	15290-129	15290-129	38400
7	124673-0	A池张沟村-HLHF-1	38400	47443-0	47443-0	38400
8	124673-0	A池张沟村-HLHF-1	38400	47444-1	47444-1	38400
9	124673-0	A池张沟村-HLHF-1	38400	124673-1	124673-1	38400

- 查询 9

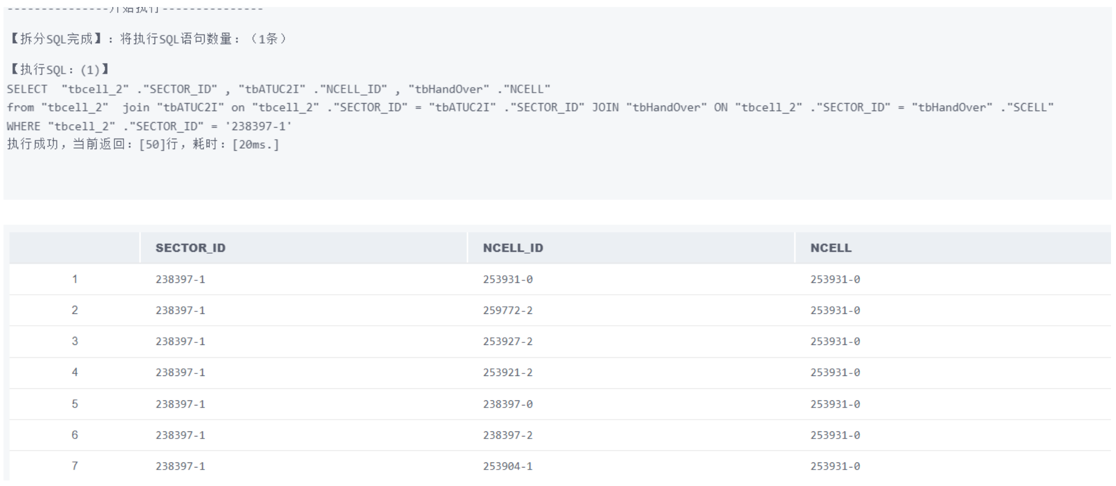
(1) 查询内容

查询 9：使用多表连接操作，从小区/基站信息表 tbCell 表、路测 ATU C2I 干扰矩阵表 tbATUC2I、路测 ATU 切换统计矩阵表 tbATUHandover 中，查询小区标识 ID 为"238397-1"的主小区的同站干扰小区的小区和切换目标小区，列出主小区名称和 ID、同站干扰小区的 ID、切换目标小区的 ID。

(2) 查询语句

```sql
SELECT "tbcell_2" ."SECTOR_ID" , "tbATUC2I" ."NCELL_ID" , "tbHandOver" ."NCELL"
from "tbcell_2"  join "tbATUC2I" on "tbcell_2" ."SECTOR_ID" = "tbATUC2I" ."SECTOR_ID" JOIN "tbHandOver" ON "tbcell_2" ."SECTOR_ID" = "tbHandOver" ."SCELL"
WHERE "tbcell_2" ."SECTOR_ID" = '238397-1'
```

(3) 查询结果

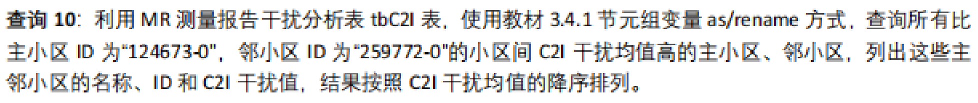


```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT  "tbcell_2" ."SECTOR_ID" , "tbATUC2I" ."NCELL_ID" , "tbHandOver" ."NCELL"
from "tbcell_2"  join "tbATUC2I" on "tbcell_2" ."SECTOR_ID" = "tbATUC2I" ."SECTOR_ID" JOIN "tbHandOver" ON "tbcell_2" ."SECTOR_ID" = "tbHandOver" ."SCELL"
WHERE "tbcell_2" ."SECTOR_ID" = '238397-1'
执行成功，当前返回：[50]行，耗时：[20ms.]
```

	SECTOR_ID	NCELL_ID	NCELL
1	238397-1	253931-0	253931-0
2	238397-1	259772-2	253931-0
3	238397-1	253927-2	253931-0
4	238397-1	253921-2	253931-0
5	238397-1	238397-0	253931-0
6	238397-1	238397-2	253931-0
7	238397-1	253904-1	253931-0

- 查询 10

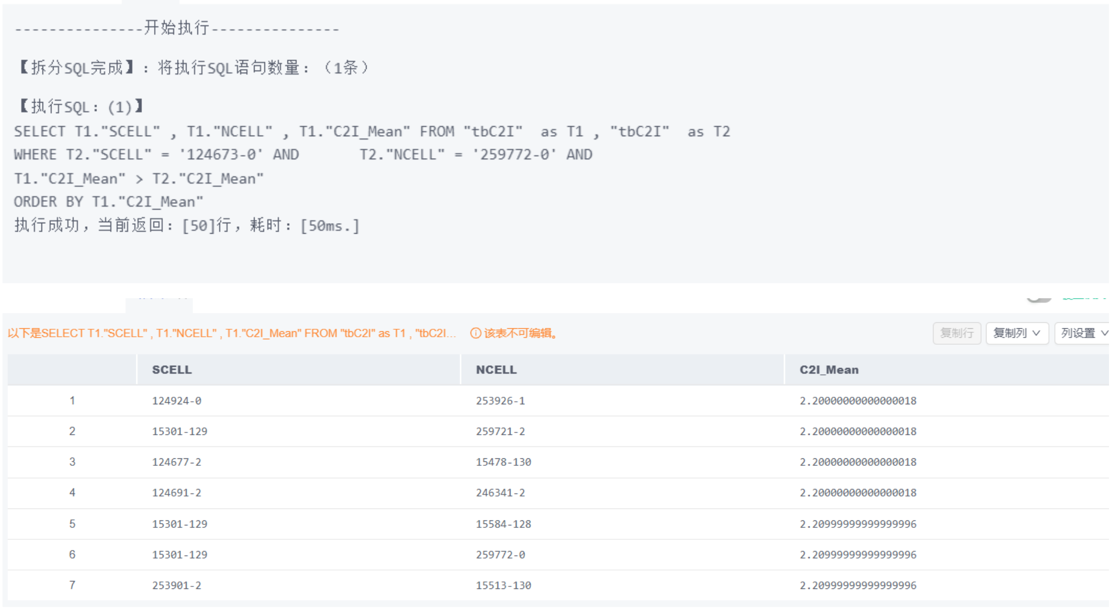
(1) 查询内容


查询 10：利用 MR 测量报告干扰分析表 tbC2I 表，使用教材 3.4.1 节元组变量 as/rename 方式，查询所有比主小区 ID 为"124673-0"，邻小区 ID 为"259772-0"的小区间 C2I 干扰均值高的主小区、邻小区，列出这些主邻小区的名称、ID 和 C2I 干扰值，结果按照 C2I 干扰均值的降序排列。

(2) 查询语句

```sql
SELECT T1."SCELL" , T1."NCELL" , T1."C2I_Mean" FROM "tbC2I"  as T1 , "tbC2I"  as T2
WHERE T2."SCELL" = '124673-0' AND   T2."NCELL" = '259772-0' AND
T1."C2I_Mean" > T2."C2I_Mean"
ORDER BY T1."C2I_Mean"
```

(3) 查询结果



```
--------------开始执行--------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT T1."SCELL" , T1."NCELL" , T1."C2I_Mean" FROM "tbC2I"  as T1 , "tbC2I"  as T2
WHERE T2."SCELL" = '124673-0' AND      T2."NCELL" = '259772-0' AND
T1."C2I_Mean" > T2."C2I_Mean"
ORDER BY T1."C2I_Mean"
执行成功，当前返回：[50]行，耗时：[50ms.]
```

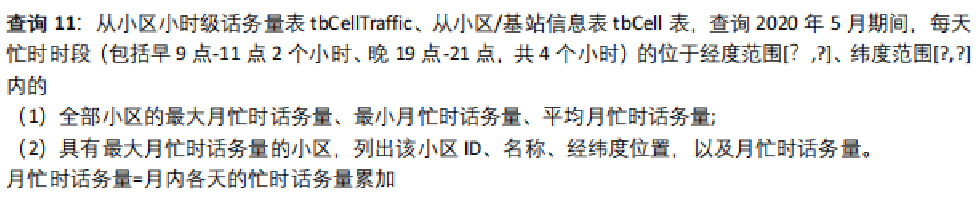
以下是SELECT T1."SCELL" , T1."NCELL" , T1."C2I_Mean" FROM "tbC2I" as T1 , "tbC2I... ⓘ 该表不可编辑。 复制行 复制列 ∨ 列设置 ∨

	SCELL	NCELL	C2I_Mean
1	124924-0	253926-1	2.20000000000000018
2	15301-129	259721-2	2.20000000000000018
3	124677-2	15478-130	2.20000000000000018
4	124691-2	246341-2	2.20000000000000018
5	15301-129	15584-128	2.20999999999999996
6	15301-129	259772-0	2.20999999999999996
7	253901-2	15513-130	2.20999999999999996

3、 统计查询

● 查询 11

(1) 查询内容


查询 11：从小区小时级话务量表 tbCellTraffic、从小区/基站信息表 tbCell 表，查询 2020 年 5 月期间，每天忙时时段（包括早 9 点-11 点 2 个小时、晚 19 点-21 点，共 4 个小时）的位于经度范围[? ,?]、纬度范围[?,?]内的

（1）全部小区的最大月忙时话务量、最小月忙时话务量、平均月忙时话务量;

（2）具有最大月忙时话务量的小区，列出该小区 ID、名称、经纬度位置，以及月忙时话务量。

月忙时话务量=月内各天的忙时话务量累加

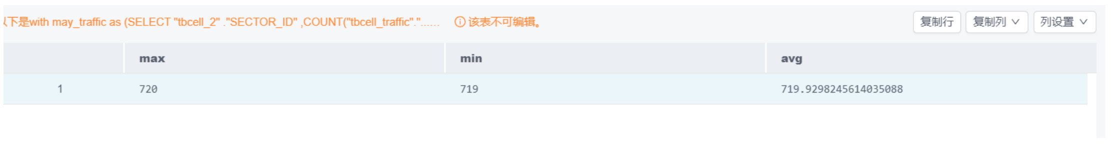
① 全部小区的最大月忙时话务量、最小月忙时话务量、平均月忙时话务量；

查询语句

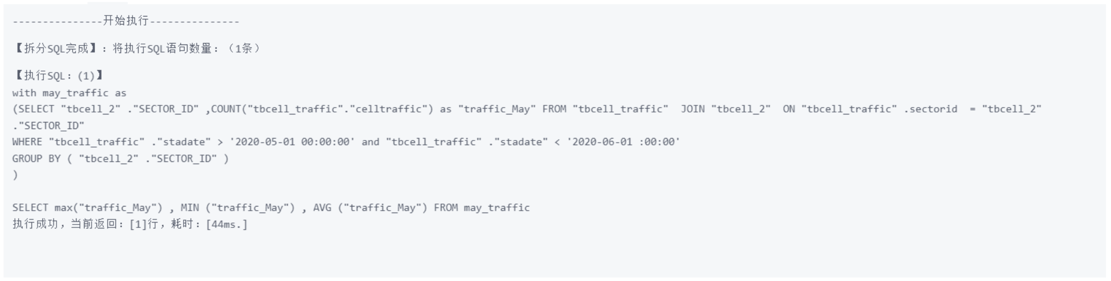
```sql
with may_traffic as
```

```
(
SELECT  "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."celltraffic") as "traffic_May" FROM "tbcell_traffic"  JOIN "tbcell_2"  ON "tbcell_traffic" .sectorid  = "tbcell_2" ."SECTOR_ID"
WHERE "tbcell_traffic" ."stadate" > '2020-05-01 00:00:00' and "tbcell_traffic" ."stadate" < '2020-06-01 :00:00'
GROUP BY ( "tbcell_2" ."SECTOR_ID" )
)

SELECT max("traffic_May") , MIN ("traffic_May") , AVG ("traffic_May") FROM may_traffic
```

查询结果


以下是with may_traffic as (SELECT "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."...... ⓘ 该表不可编辑。 复制行 复制列 列设置

	max	min	avg
1	720	719	719.9298245614035088

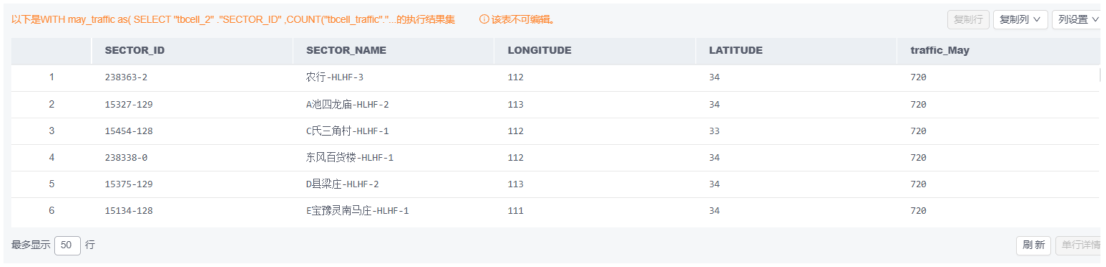

```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
with may_traffic as
(SELECT "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."celltraffic") as "traffic_May" FROM "tbcell_traffic"  JOIN "tbcell_2"  ON "tbcell_traffic" .sectorid  = "tbcell_2"
."SECTOR_ID"
WHERE "tbcell_traffic" ."stadate" > '2020-05-01 00:00:00' and "tbcell_traffic" ."stadate" < '2020-06-01 :00:00'
GROUP BY ( "tbcell_2" ."SECTOR_ID" )
)

SELECT max("traffic_May") , MIN ("traffic_May") , AVG ("traffic_May") FROM may_traffic
执行成功，当前返回：[1]行，耗时：[44ms.]
```

② 具有最大月忙时话务量的小区，列出该小区的 ID，名称，经纬度，月忙时话务量

查询语句

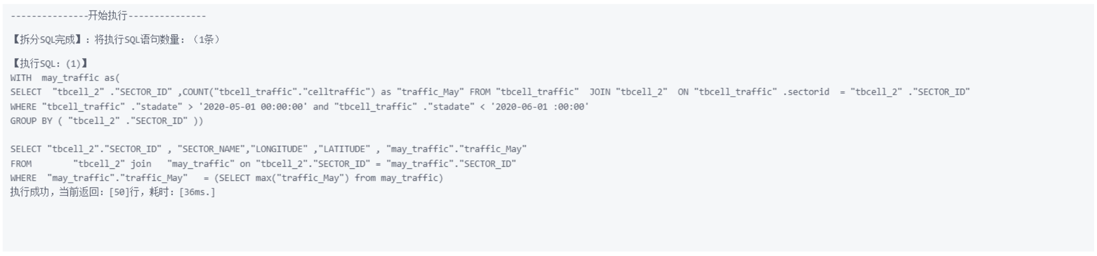
```
WITH  may_traffic as(
SELECT  "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."celltraffic") as "traffic_May" FROM "tbcell_traffic"  JOIN "tbcell_2"  ON "tbcell_traffic" .sectorid  = "tbcell_2" ."SECTOR_ID"
WHERE "tbcell_traffic" ."stadate" > '2020-05-01 00:00:00' and "tbcell_traffic" ."stadate" < '2020-06-01 :00:00'
GROUP BY ( "tbcell_2" ."SECTOR_ID" ))

SELECT "tbcell_2"."SECTOR_ID" , "SECTOR_NAME","LONGITUDE" ,"LATITUDE" , "may_traffic"."traffic_May"

FROM        "tbcell_2" join   "may_traffic" on "tbcell_2"."SECTOR_ID" = "may_traffic"."SECTOR_ID"
WHERE  "may_traffic"."traffic_May"   = (SELECT max("traffic_May") from may_traffic)
```

查询结果


以下是WITH may_traffic as(SELECT "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."...的执行结果集 ⓘ 该表不可编辑。 复制行 复制列 列设置

	SECTOR_ID	SECTOR_NAME	LONGITUDE	LATITUDE	traffic_May
1	238363-2	农行-HLHF-3	112	34	720
2	15327-129	A池四龙庙-HLHF-2	113	34	720
3	15454-128	C氏三角村-HLHF-1	112	33	720
4	238338-0	东风百货楼-HLHF-1	112	34	720
5	15375-129	D县梁庄-HLHF-2	113	34	720
6	15134-128	E宝豫灵南马庄-HLHF-1	111	34	720

最多显示 50 行 刷新 单行详情



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
WITH  may_traffic as(
SELECT  "tbcell_2" ."SECTOR_ID" ,COUNT("tbcell_traffic"."celltraffic") as "traffic_May" FROM "tbcell_traffic"  JOIN "tbcell_2"  ON "tbcell_traffic" .sectorid  = "tbcell_2" ."SECTOR_ID"
WHERE "tbcell_traffic" ."stadate" > '2020-05-01 00:00:00' and "tbcell_traffic" ."stadate" < '2020-06-01 :00:00'
GROUP BY ( "tbcell_2" ."SECTOR_ID" ))

SELECT "tbcell_2"."SECTOR_ID" , "SECTOR_NAME","LONGITUDE" ,"LATITUDE" , "may_traffic"."traffic_May"
FROM        "tbcell_2" join   "may_traffic" on "tbcell_2"."SECTOR_ID" = "may_traffic"."SECTOR_ID"
WHERE  "may_traffic"."traffic_May"   = (SELECT max("traffic_May") from may_traffic)
执行成功，当前返回：[50]行，耗时：[36ms.]
```

- 查询 12

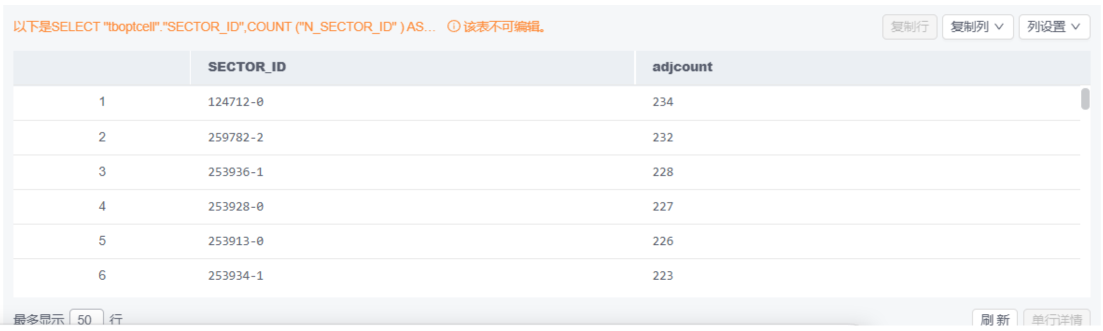
(1) 查询内容

查询 12：根据优化小区/保护带小区表 tbOptCell 和小区一阶邻区关系表 tbAdjCell，查询一阶邻区数大于 10 的优化小区，给出这些优化小区的标识、名称，以及邻区数量，并将查询结果按照邻区数目降序排列。

(2) 查询语句

```sql
SELECT "tboptcell"."SECTOR_ID",COUNT ("N_SECTOR_ID" ) AS ADJCOUNT  FROM "tbAdjCell" JOIN "tboptcell"
 on "tboptcell"."SECTOR_ID" = "tbAdjCell"."S_SECTOR_ID"
  GROUP BY "tboptcell"."SECTOR_ID"
  HAVING (ADJCOUNT>10)
ORDER BY ADJCOUNT DESC

```

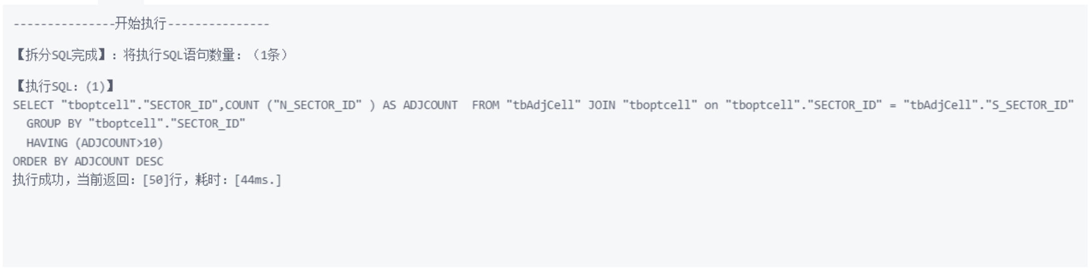
(3) 查询结果



以下是SELECT "tboptcell"."SECTOR_ID",COUNT ("N_SECTOR_ID") AS... ⓘ 该表不可编辑。

	SECTOR_ID	adjcount
1	124712-0	234
2	259782-2	232
3	253936-1	228
4	253928-0	227
5	253913-0	226
6	253934-1	223

最多显示 50 行



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT "tboptcell"."SECTOR_ID",COUNT ("N_SECTOR_ID" ) AS ADJCOUNT  FROM "tbAdjCell" JOIN "tboptcell" on "tboptcell"."SECTOR_ID" = "tbAdjCell"."S_SECTOR_ID"
  GROUP BY "tboptcell"."SECTOR_ID"
  HAVING (ADJCOUNT>10)
ORDER BY ADJCOUNT DESC
执行成功，当前返回：[50]行，耗时：[44ms.]
```

- ·查询 13

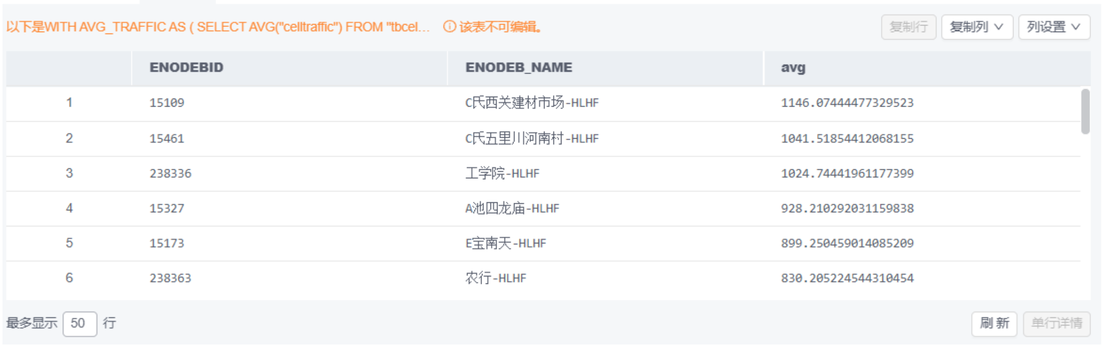
(1)　查询内容

查询 13：从小区话务量表 tbCellTraffic、小区/基站信息表 tbCell 表中，查询所有包含 38400 频点的基站的年平均小时级话务量 avgTraffic，给出年平均话务量超出 avgTraffic 的基站 ID 名称、基站年平均话务量，结果按照年平均话务量降序排列。

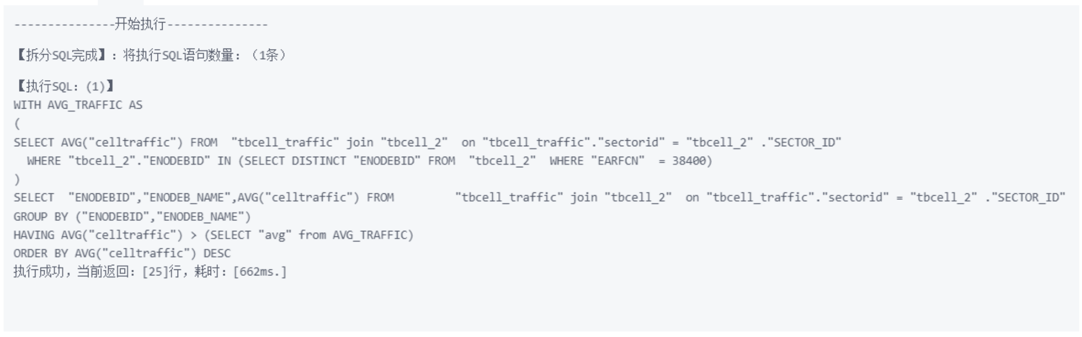
（2） 查询语句

```sql
WITH AVG_TRAFFIC AS
(
SELECT AVG("celltraffic") FROM  "tbcell_traffic" join "tbcell_2"  on "tbcell_traffic"."sectorid" = "tbcell_2" ."SECTOR_ID"
  WHERE "tbcell_2"."ENODEBID" IN (SELECT DISTINCT "ENODEBID" FROM  "tbcell_2"  WHERE "EARFCN"  = 38400)
)
SELECT  "ENODEBID","ENODEB_NAME",AVG("celltraffic") FROM     "tbcell_traffic" join "tbcell_2"  on "tbcell_traffic"."sectorid" = "tbcell_2" ."SECTOR_ID"
GROUP BY ("ENODEBID","ENODEB_NAME")
HAVING AVG("celltraffic") > (SELECT "avg" from AVG_TRAFFIC)
ORDER BY AVG("celltraffic") DESC
```

（3） 查询结果


以下是WITH AVG_TRAFFIC AS (SELECT AVG("celltraffic") FROM "tbcel...
ⓘ 该表不可编辑。
复制行
复制列
列设置
最多显示 50 行
刷 新
单行详情

	ENODEBID	ENODEB_NAME	avg
1	15109	C氏西关建材市场-HLHF	1146.07444477329523
2	15461	C氏五里川河南村-HLHF	1041.51854412068155
3	238336	工学院-HLHF	1024.74441961177399
4	15327	A池四龙庙-HLHF	928.210292031159838
5	15173	E宝南天-HLHF	899.250459014085209
6	238363	农行-HLHF	830.205224544310454



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
WITH AVG_TRAFFIC AS
(
SELECT AVG("celltraffic") FROM  "tbcell_traffic" join "tbcell_2"  on "tbcell_traffic"."sectorid" = "tbcell_2" ."SECTOR_ID"
  WHERE "tbcell_2"."ENODEBID" IN (SELECT DISTINCT "ENODEBID" FROM  "tbcell_2"  WHERE "EARFCN"  = 38400)
)
SELECT  "ENODEBID","ENODEB_NAME",AVG("celltraffic") FROM         "tbcell_traffic" join "tbcell_2"  on "tbcell_traffic"."sectorid" = "tbcell_2" ."SECTOR_ID"
GROUP BY ("ENODEBID","ENODEB_NAME")
HAVING AVG("celltraffic") > (SELECT "avg" from AVG_TRAFFIC)
ORDER BY AVG("celltraffic") DESC
执行成功，当前返回：[25]行，耗时：[662ms.]
```

4、 嵌套查询

● 查询 14

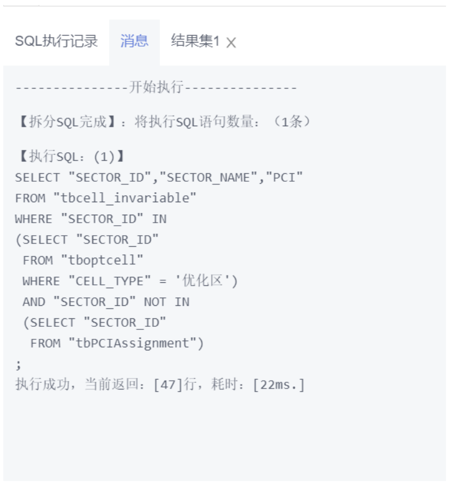
（1）查询内容

查询 15-1：从小区/基站信息表 tbCell，使用 Set Comparison 运算符 some，查询满足下列条件的小区：该小区的天线高度 height 高于位于经度在[?,?]、纬度在[?,?]区域内的部分（至少一个）小区的天线高度，列出这些小区的名称、标识和天线高度。

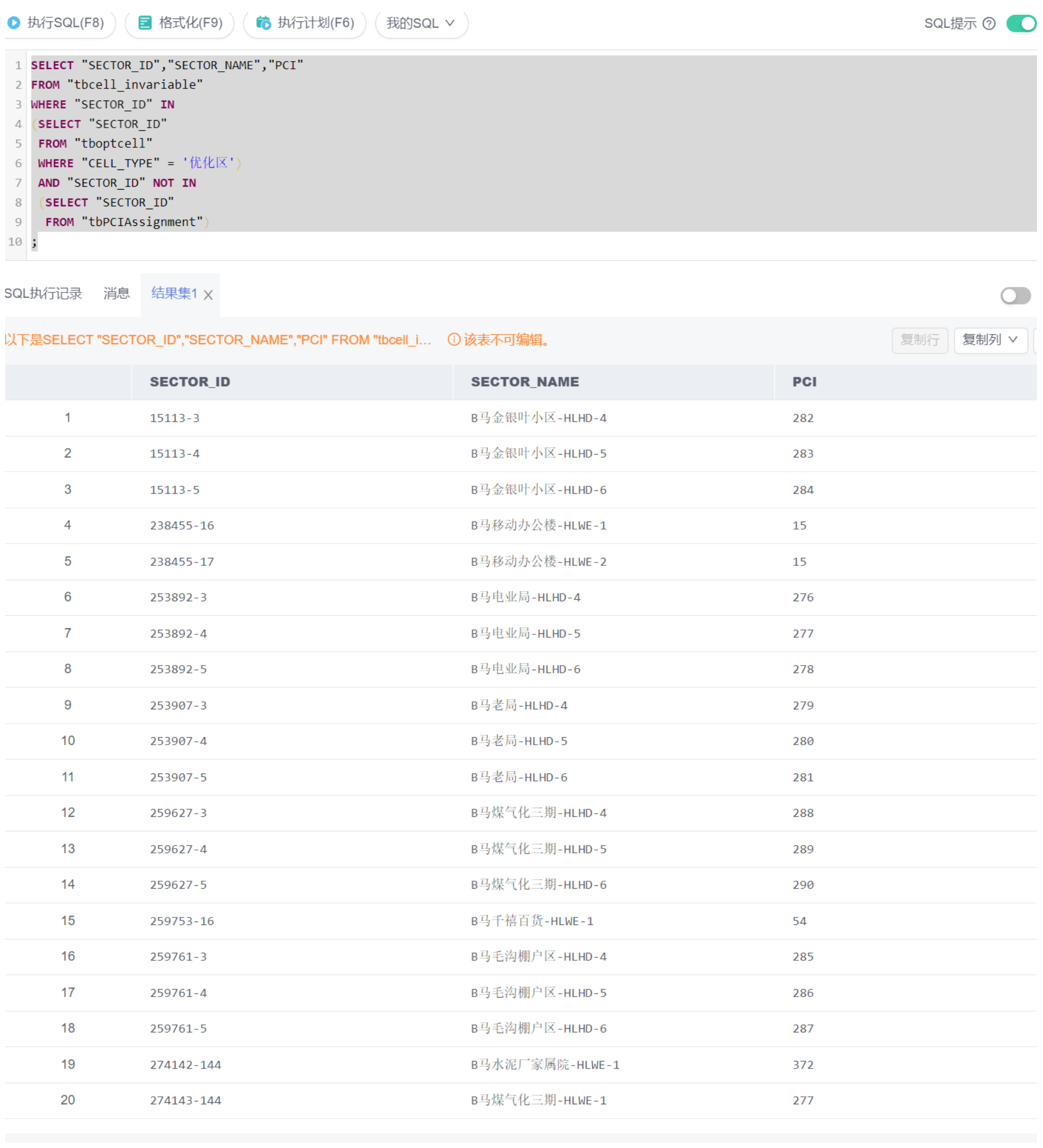
（2）查询语句

```
SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT"
FROM "tbcell_invariable"
WHERE "HEIGHT" > SOME
(SELECT "HEIGHT"
 FROM "tbcell_invariable"
 WHERE "LONGITUDE" BETWEEN 111.5 AND 112.5
 AND "LATITUDE" BETWEEN 33.5 AND 34.5)
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: （1条）

【执行SQL: (1)】
SELECT "SECTOR_ID","SECTOR_NAME","PCI"
FROM "tbcell_invariable"
WHERE "SECTOR_ID" IN
(SELECT "SECTOR_ID"
 FROM "tboptcell"
 WHERE "CELL_TYPE" = '优化区')
 AND "SECTOR_ID" NOT IN
 (SELECT "SECTOR_ID"
  FROM "tbPCIAssignment")
;
执行成功，当前返回：[47]行，耗时：[22ms.]
```



```
SELECT "SECTOR_ID","SECTOR_NAME","PCI"
FROM "tbcell_invariable"
WHERE "SECTOR_ID" IN
(SELECT "SECTOR_ID"
 FROM "tboptcell"
 WHERE "CELL_TYPE" = '优化区')
 AND "SECTOR_ID" NOT IN
 (SELECT "SECTOR_ID"
  FROM "tbPCIAssignment")
;
```

以下是SELECT "SECTOR_ID","SECTOR_NAME","PCI" FROM "tbcell_i… ⓘ 该表不可编辑。

	SECTOR_ID	SECTOR_NAME	PCI
1	15113-3	B马金银叶小区-HLHD-4	282
2	15113-4	B马金银叶小区-HLHD-5	283
3	15113-5	B马金银叶小区-HLHD-6	284
4	238455-16	B马移动办公楼-HLWE-1	15
5	238455-17	B马移动办公楼-HLWE-2	15
6	253892-3	B马电业局-HLHD-4	276
7	253892-4	B马电业局-HLHD-5	277
8	253892-5	B马电业局-HLHD-6	278
9	253907-3	B马老局-HLHD-4	279
10	253907-4	B马老局-HLHD-5	280
11	253907-5	B马老局-HLHD-6	281
12	259627-3	B马煤气化三期-HLHD-4	288
13	259627-4	B马煤气化三期-HLHD-5	289
14	259627-5	B马煤气化三期-HLHD-6	290
15	259753-16	B马千禧百货-HLWE-1	54
16	259761-3	B马毛沟棚户区-HLHD-4	285
17	259761-4	B马毛沟棚户区-HLHD-5	286
18	259761-5	B马毛沟棚户区-HLHD-6	287
19	274142-144	B马水泥厂家属院-HLWE-1	372
20	274143-144	B马煤气化三期-HLWE-1	277

- 查询 15-1

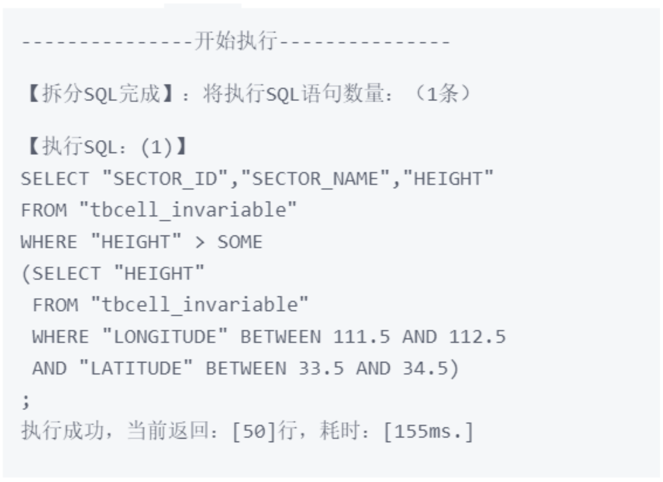
（1）查询内容

查询 15-1：从小区/基站信息表 tbCell，使用 Set Comparison 运算符 some，查询满足下列条件的小区：该小区的天线高度 height 高于位于经度在[?,?]、纬度在[?,?]区域内的部分（至少一个）小区的天线高度，列出这些小区的名称、标识和天线高度。

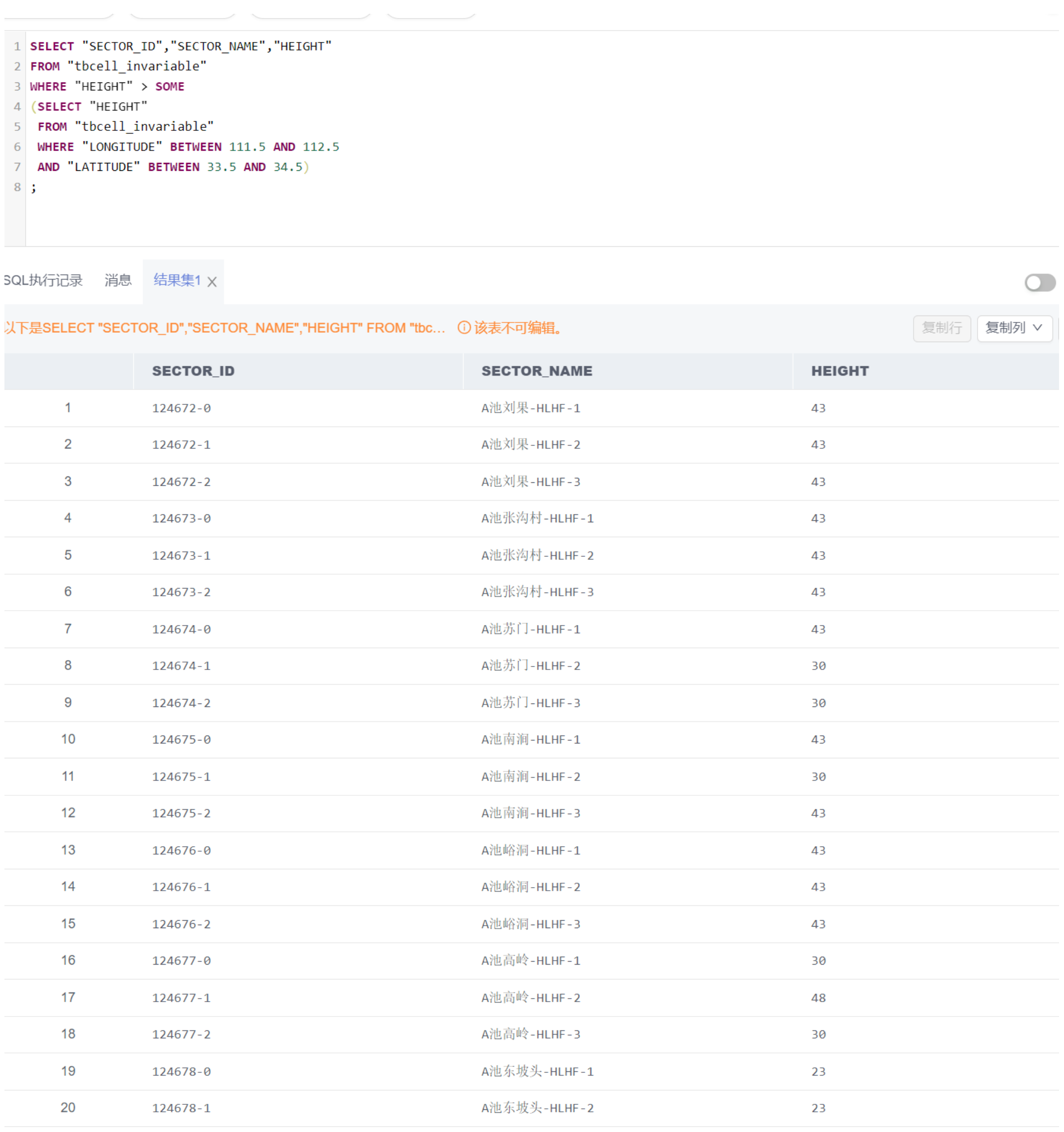
（2）查询语句

```
SELECT "SECTOR_ID","SECTOR_NAME","PCI"
SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT"
FROM "tbcell_invariable"
WHERE "HEIGHT" > SOME
(SELECT "HEIGHT"
 FROM "tbcell_invariable"
 WHERE "LONGITUDE" BETWEEN 111.5 AND 112.5
 AND "LATITUDE" BETWEEN 33.5 AND 34.5)
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT"
FROM "tbcell_invariable"
WHERE "HEIGHT" > SOME
(SELECT "HEIGHT"
 FROM "tbcell_invariable"
 WHERE "LONGITUDE" BETWEEN 111.5 AND 112.5
 AND "LATITUDE" BETWEEN 33.5 AND 34.5)
;
执行成功，当前返回：[50]行，耗时：[155ms.]
```



```sql
SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT"
FROM "tbcell_invariable"
WHERE "HEIGHT" > SOME
(SELECT "HEIGHT"
 FROM "tbcell_invariable"
 WHERE "LONGITUDE" BETWEEN 111.5 AND 112.5
 AND "LATITUDE" BETWEEN 33.5 AND 34.5)
;
```

以下是SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT" FROM "tbc... ⓘ该表不可编辑。

	SECTOR_ID	SECTOR_NAME	HEIGHT
1	124672-0	A池刘果-HLHF-1	43
2	124672-1	A池刘果-HLHF-2	43
3	124672-2	A池刘果-HLHF-3	43
4	124673-0	A池张沟村-HLHF-1	43
5	124673-1	A池张沟村-HLHF-2	43
6	124673-2	A池张沟村-HLHF-3	43
7	124674-0	A池苏门-HLHF-1	43
8	124674-1	A池苏门-HLHF-2	30
9	124674-2	A池苏门-HLHF-3	30
10	124675-0	A池南涧-HLHF-1	43
11	124675-1	A池南涧-HLHF-2	30
12	124675-2	A池南涧-HLHF-3	43
13	124676-0	A池峪涧-HLHF-1	43
14	124676-1	A池峪涧-HLHF-2	43
15	124676-2	A池峪涧-HLHF-3	43
16	124677-0	A池高岭-HLHF-1	30
17	124677-1	A池高岭-HLHF-2	48
18	124677-2	A池高岭-HLHF-3	30
19	124678-0	A池东坡头-HLHF-1	23
20	124678-1	A池东坡头-HLHF-2	23

- 查询 15-2

（1）查询内容

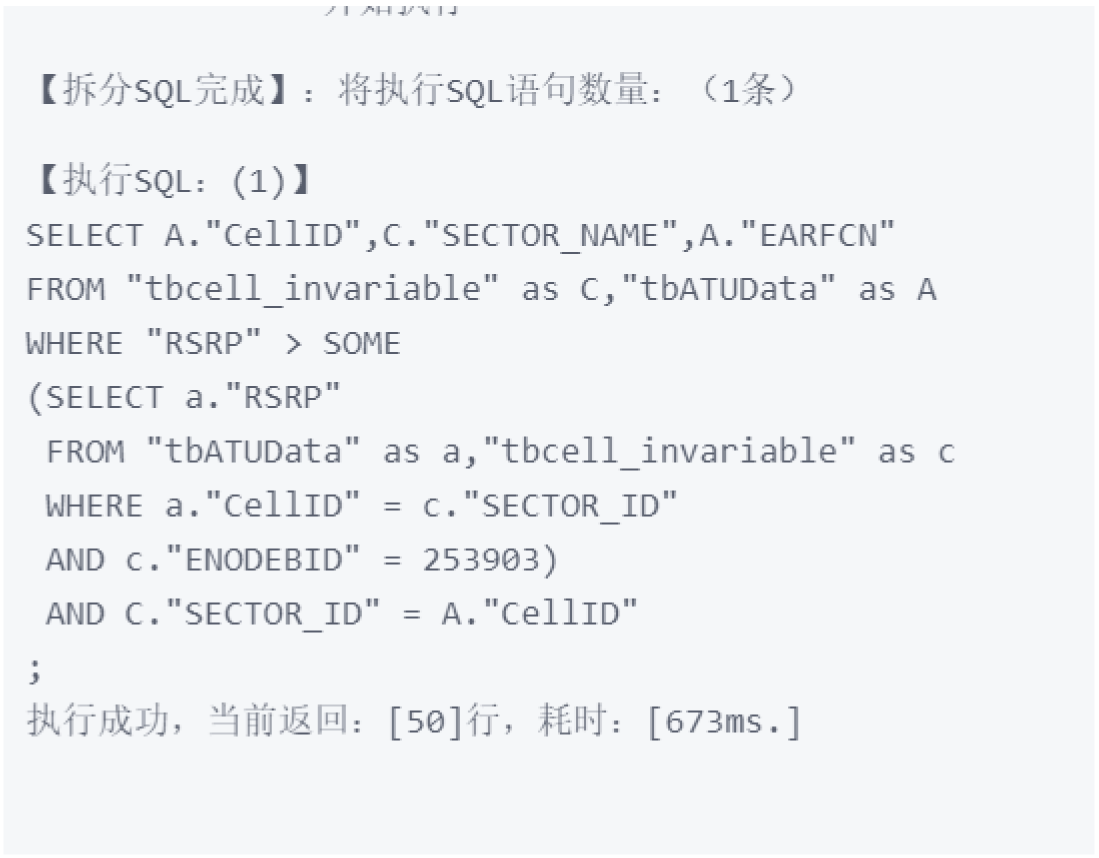
查询 15-2：从路测 ATU 数据表 tbATUdata，使用 Set Comparison 运算符 some，查询满足下列条件的小区：在路测数据中作为主小区/服务小区 CELLID，其参考信号接收功率 RSRP，大于部分（基站标识 ENodeBID=253903 的小区作为主小区/服务小区时的参考信号接收功率 RSRP。列出这些小区的 ID、名称、在测量报告中作为主服务小区的 RSRP。

（2）查询语句

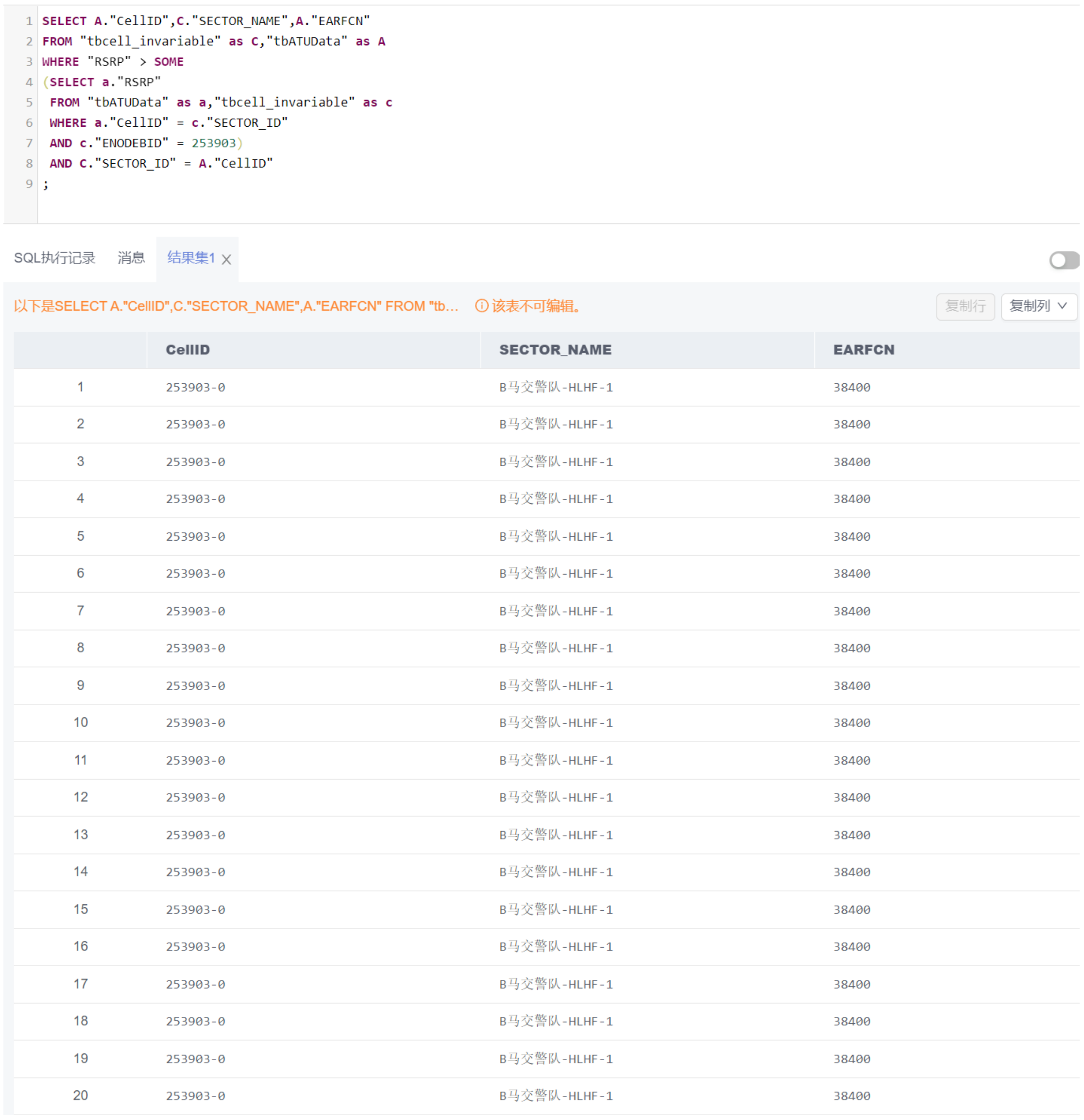
建表：tbATUdata

```sql
create table "tbATUData"
(
  "seq" BIGINT ,
  "FileName" VARCHAR (255),
  "Time" VARCHAR (100),
  "Longitude" float,
  "Latitude" float,
  "CellID" VARCHAR (50),
  "TAC" int,
  "EARFCN" int,
  "PCI" SMALLINT ,
  "RSRP" float,
  "RS_SINR" float,
  "NCell_ID_1" VARCHAR (50),
  "NCell_EARFCN_1" int,
  "NCell_PCI_1" SMALLINT ,
  "NCell_RSRP_1" float,
  "NCell_ID_2" VARCHAR (50),
  "NCell_EARFCN_2" int,
  "NCell_PCI_2" SMALLINT ,
  "NCell_RSRP_2" float,
  "NCell_ID_3" VARCHAR (50),
  "NCell_EARFCN_3" int,
  "NCell_PCI_3" SMALLINT ,
  "NCell_RSRP_3" float,
  "NCell_ID_4" VARCHAR (50),
  "NCell_EARFCN_4" int,
  "NCell_PCI_4" SMALLINT ,
  "NCell_RSRP_4" float,
  "NCell_ID_5" VARCHAR (50),
  "NCell_EARFCN_5" int,
  "NCell_PCI_5" SMALLINT ,
  "NCell_RSRP_5" float,
  "NCell_ID_6" VARCHAR (50),
  "NCell_EARFCN_6" int,
  "NCell_PCI_6" SMALLINT ,
  "NCell_RSRP_6" float,
  primary key ("seq", "FileName")
);
```
查询：
```
SELECT A."CellID",C."SECTOR_NAME",A."EARFCN"
FROM "tbcell_invariable" as C,"tbATUData" as A
WHERE "RSRP" > SOME
(SELECT a."RSRP"
 FROM "tbATUData" as a,"tbcell_invariable" as c
 WHERE a."CellID" = c."SECTOR_ID"
 AND c."ENODEBID" = 253903)
 AND C."SECTOR_ID" = A."CellID"
;
```

（3）查询结果



```
【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT A."CellID",C."SECTOR_NAME",A."EARFCN"
FROM "tbcell_invariable" as C,"tbATUData" as A
WHERE "RSRP" > SOME
(SELECT a."RSRP"
 FROM "tbATUData" as a,"tbcell_invariable" as c
 WHERE a."CellID" = c."SECTOR_ID"
 AND c."ENODEBID" = 253903)
 AND C."SECTOR_ID" = A."CellID"
;
执行成功，当前返回：[50]行，耗时：[673ms.]
```



```sql
SELECT A."CellID",C."SECTOR_NAME",A."EARFCN"
FROM "tbcell_invariable" as C,"tbATUData" as A
WHERE "RSRP" > SOME
(SELECT a."RSRP"
 FROM "tbATUData" as a,"tbcell_invariable" as c
 WHERE a."CellID" = c."SECTOR_ID"
 AND c."ENODEBID" = 253903)
 AND C."SECTOR_ID" = A."CellID"
;
```

以下是SELECT A."CellID",C."SECTOR_NAME",A."EARFCN" FROM "tb… ⓘ该表不可编辑。

	CellID	SECTOR_NAME	EARFCN
1	253903-0	B马交警队-HLHF-1	38400
2	253903-0	B马交警队-HLHF-1	38400
3	253903-0	B马交警队-HLHF-1	38400
4	253903-0	B马交警队-HLHF-1	38400
5	253903-0	B马交警队-HLHF-1	38400
6	253903-0	B马交警队-HLHF-1	38400
7	253903-0	B马交警队-HLHF-1	38400
8	253903-0	B马交警队-HLHF-1	38400
9	253903-0	B马交警队-HLHF-1	38400
10	253903-0	B马交警队-HLHF-1	38400
11	253903-0	B马交警队-HLHF-1	38400
12	253903-0	B马交警队-HLHF-1	38400
13	253903-0	B马交警队-HLHF-1	38400
14	253903-0	B马交警队-HLHF-1	38400
15	253903-0	B马交警队-HLHF-1	38400
16	253903-0	B马交警队-HLHF-1	38400
17	253903-0	B马交警队-HLHF-1	38400
18	253903-0	B马交警队-HLHF-1	38400
19	253903-0	B马交警队-HLHF-1	38400
20	253903-0	B马交警队-HLHF-1	38400

- 查询 16-1

（1）查询内容

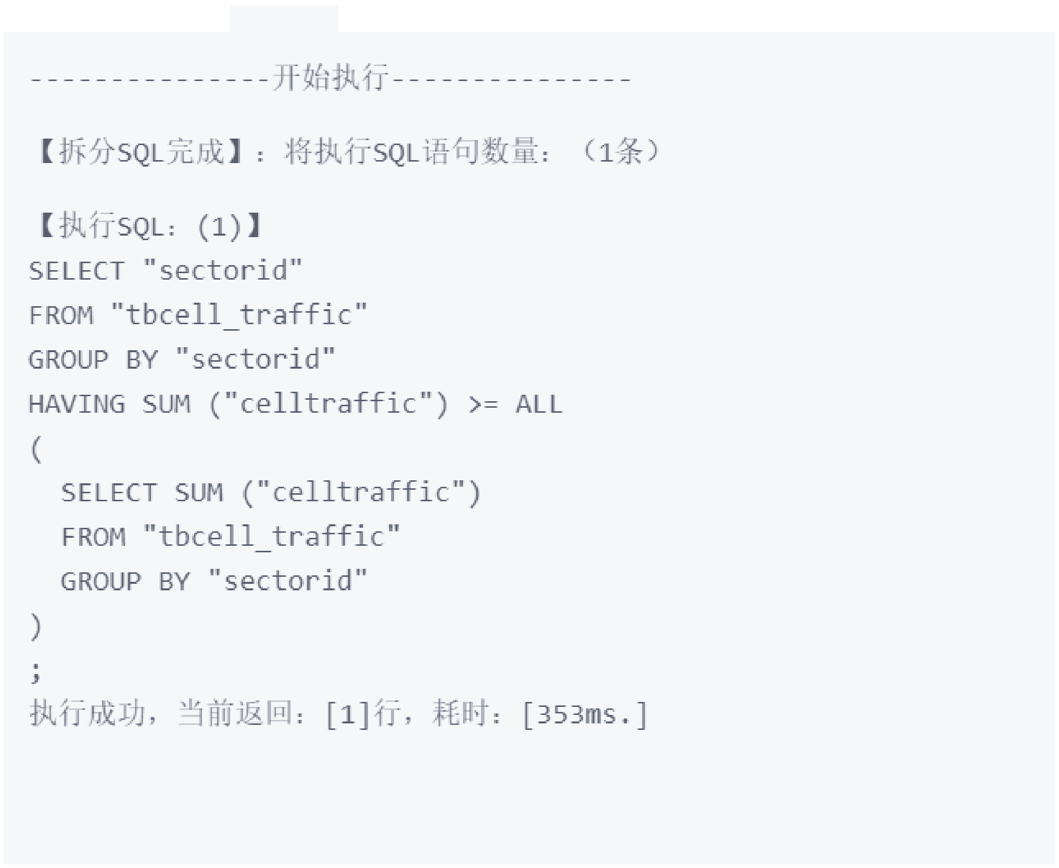
查询 16-1：从小区小时级话务量表 tbCellTraffic 中，使用 Set Comparison 运算符>=all，查询全年小时级话务量总和满足下列条件的小区：该小区的全年话务量总和大于等于其它小区的全年话务量总和，即该小区的全年话务量总和最高。

（2）查询语句

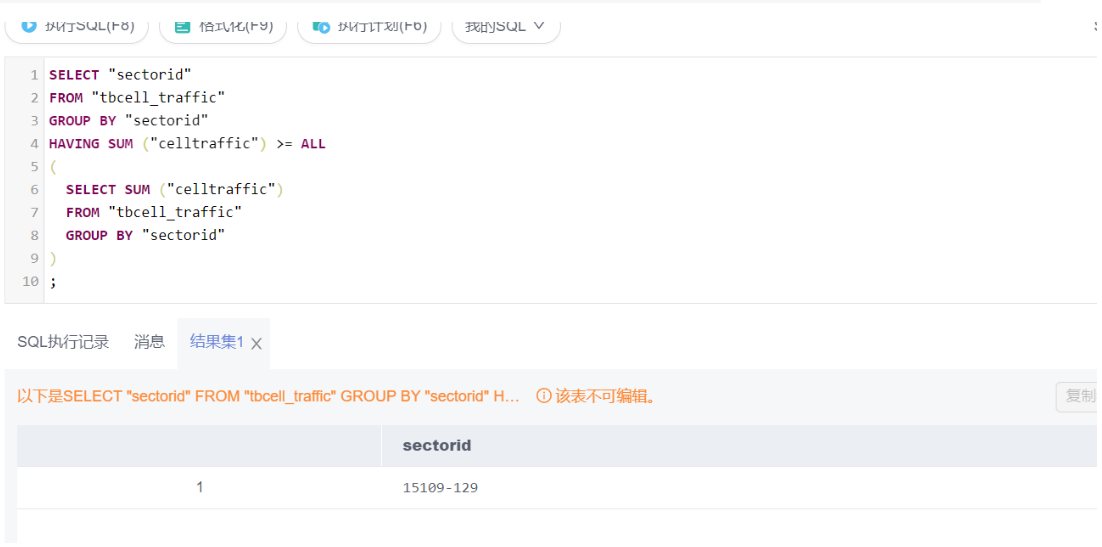
```sql
SELECT "sectorid"
FROM "tbcell_traffic"
GROUP BY "sectorid"
HAVING SUM ("celltraffic") >= ALL
(
  SELECT SUM ("celltraffic")
  FROM "tbcell_traffic"
  GROUP BY "sectorid"
)
```

```
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: （1条）

【执行SQL: (1)】
SELECT "sectorid"
FROM "tbcell_traffic"
GROUP BY "sectorid"
HAVING SUM ("celltraffic") >= ALL
(
  SELECT SUM ("celltraffic")
  FROM "tbcell_traffic"
  GROUP BY "sectorid"
)
;
执行成功, 当前返回: [1]行, 耗时: [353ms.]
```



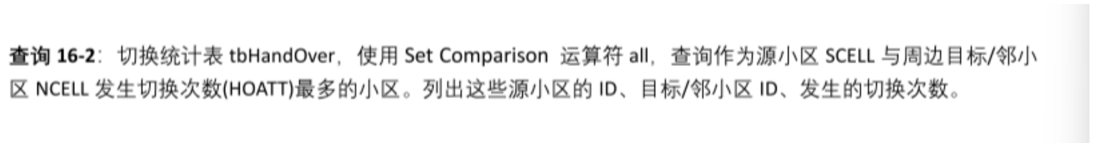
```
SELECT "sectorid"
FROM "tbcell_traffic"
GROUP BY "sectorid"
HAVING SUM ("celltraffic") >= ALL
(
  SELECT SUM ("celltraffic")
  FROM "tbcell_traffic"
  GROUP BY "sectorid"
)
;
```

以下是SELECT "sectorid" FROM "tbcell_traffic" GROUP BY "sectorid" H... ⓘ该表不可编辑。

	sectorid
1	15109-129

● 查询 16-2

（1）查询内容

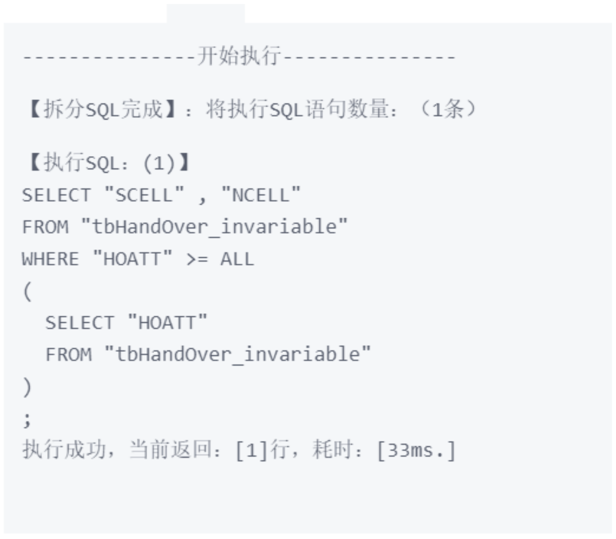


查询 16-2：切换统计表 tbHandOver，使用 Set Comparison 运算符 all，查询作为源小区 SCELL 与周边目标/邻小区 NCELL 发生切换次数(HOATT)最多的小区。列出这些源小区的 ID、目标/邻小区 ID、发生的切换次数。

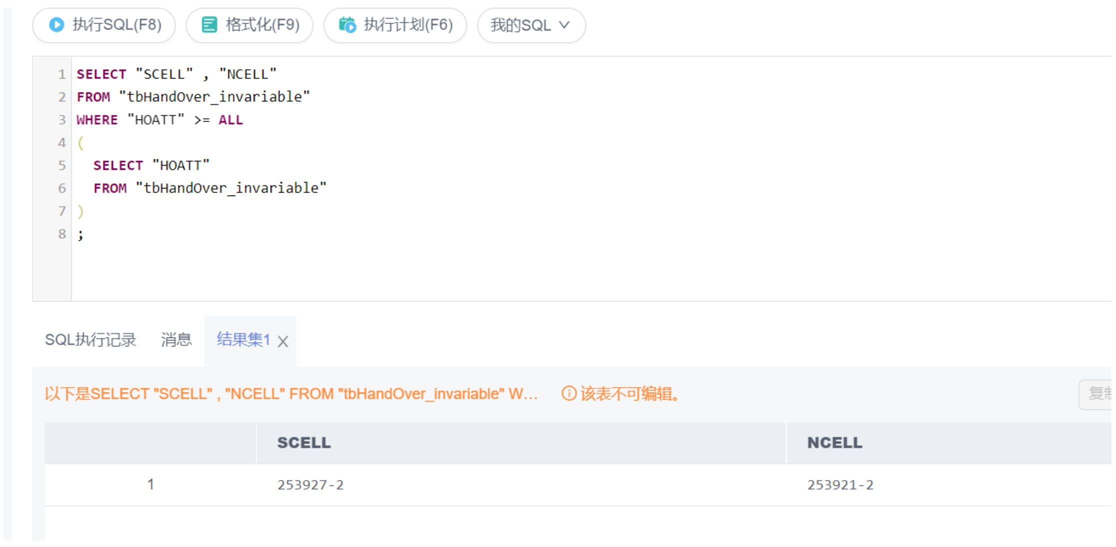
（2）查询语句

```sql
SELECT "SCELL" , "NCELL"
FROM "tbHandOver_invariable"
WHERE "HOATT" >= ALL
(
  SELECT "HOATT"
  FROM "tbHandOver_invariable"
)
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT "SCELL" , "NCELL"
FROM "tbHandOver_invariable"
WHERE "HOATT" >= ALL
(
  SELECT "HOATT"
  FROM "tbHandOver_invariable"
)
;
执行成功，当前返回：[1]行，耗时：[33ms.]
```



```sql
SELECT "SCELL" , "NCELL"
FROM "tbHandOver_invariable"
WHERE "HOATT" >= ALL
(
  SELECT "HOATT"
  FROM "tbHandOver_invariable"
)
;
```

以下是SELECT "SCELL" , "NCELL" FROM "tbHandOver_invariable" W… ⓘ 该表不可编辑。 复

	SCELL	NCELL
1	253927-2	253921-2

● 查询 17-1

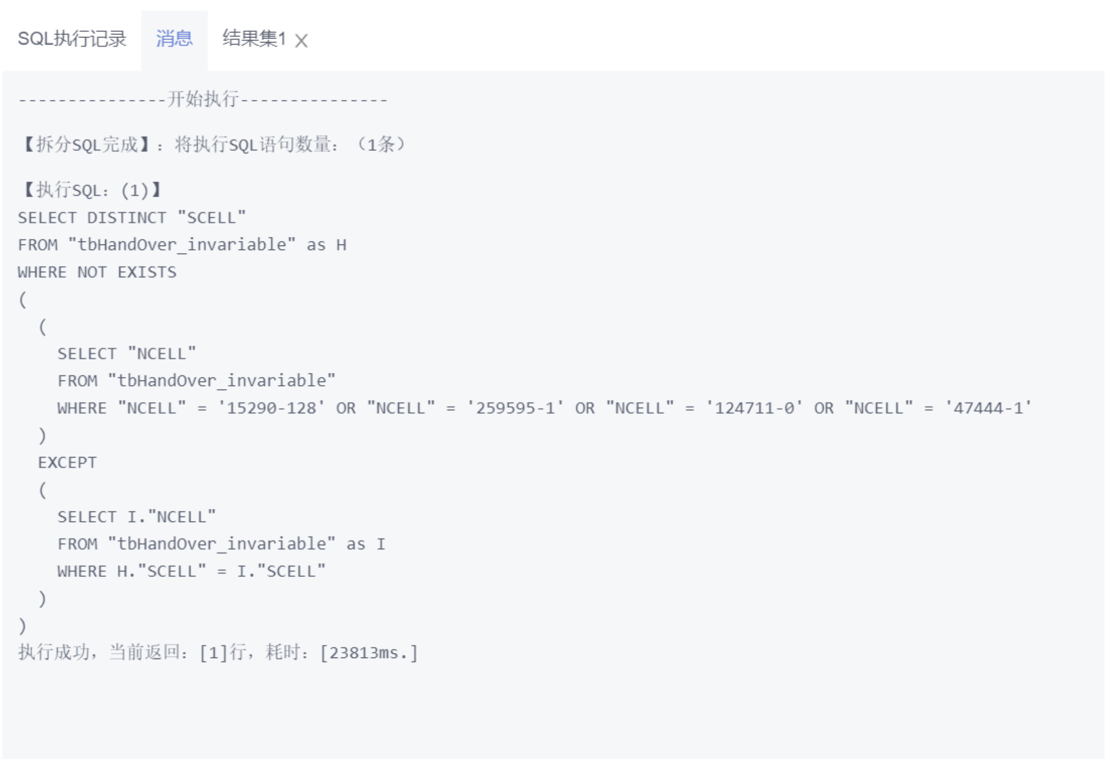
（1）查询内容

查询 17-1：从切换统计表 tbHandOver 表，使用 Test for Empty Relations 运算符“not exists”，查询作为源小区 SCELL，其切换邻小区 NCELL 包含了{15290-128, 259595-1, 124711-0, 47444-1}中的全部四个小区。

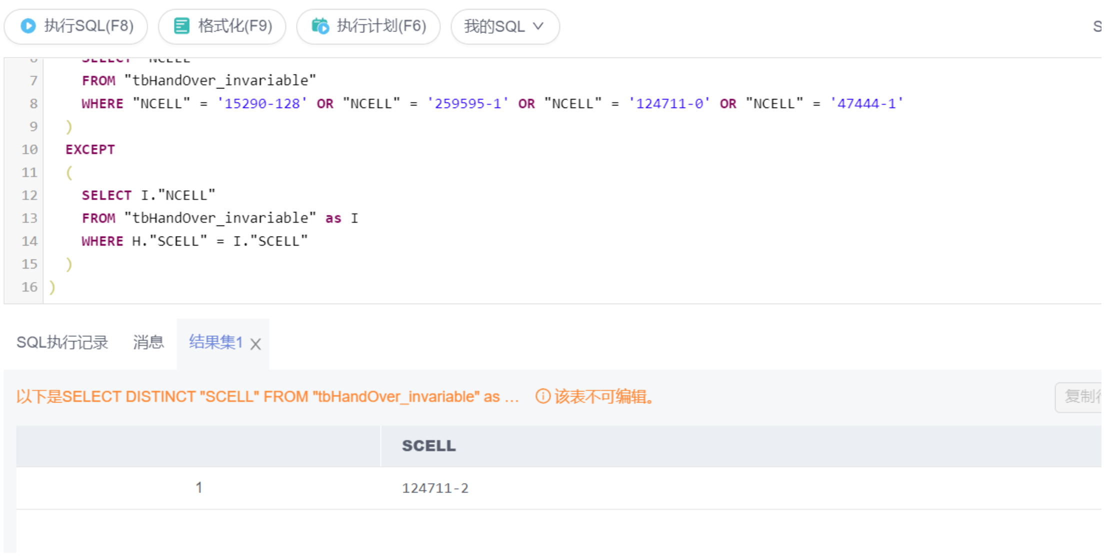
（2）查询语句

```
SELECT DISTINCT "SCELL"
FROM "tbHandOver_invariable" as H
WHERE NOT EXISTS
(
  (
    SELECT "NCELL"
    FROM "tbHandOver_invariable"
    WHERE "NCELL" = '15290-128' OR "NCELL" = '259595-1' OR "NCELL" =
'124711-0' OR "NCELL" = '47444-1'
  )
  EXCEPT
  (
    SELECT I."NCELL"
    FROM "tbHandOver_invariable" as I
    WHERE H."SCELL" = I."SCELL"
  )
)
```

（3）查询结果



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT DISTINCT "SCELL"
FROM "tbHandOver_invariable" as H
WHERE NOT EXISTS
(
  (
    SELECT "NCELL"
    FROM "tbHandOver_invariable"
    WHERE "NCELL" = '15290-128' OR "NCELL" = '259595-1' OR "NCELL" = '124711-0' OR "NCELL" = '47444-1'
  )
  EXCEPT
  (
    SELECT I."NCELL"
    FROM "tbHandOver_invariable" as I
    WHERE H."SCELL" = I."SCELL"
  )
)
执行成功，当前返回：[1]行，耗时：[23813ms.]
```



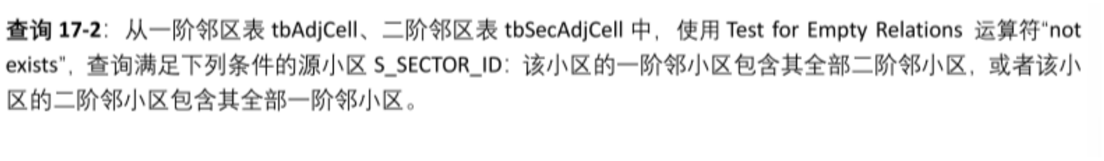
```
    FROM "tbHandOver_invariable"
    WHERE "NCELL" = '15290-128' OR "NCELL" = '259595-1' OR "NCELL" = '124711-0' OR "NCELL" = '47444-1'
  )
  EXCEPT
  (
    SELECT I."NCELL"
    FROM "tbHandOver_invariable" as I
    WHERE H."SCELL" = I."SCELL"
  )
)
```

以下是SELECT DISTINCT "SCELL" FROM "tbHandOver_invariable" as ... ⓘ该表不可编辑。

	SCELL
1	124711-2

● 查询 17-2

（1）查询内容

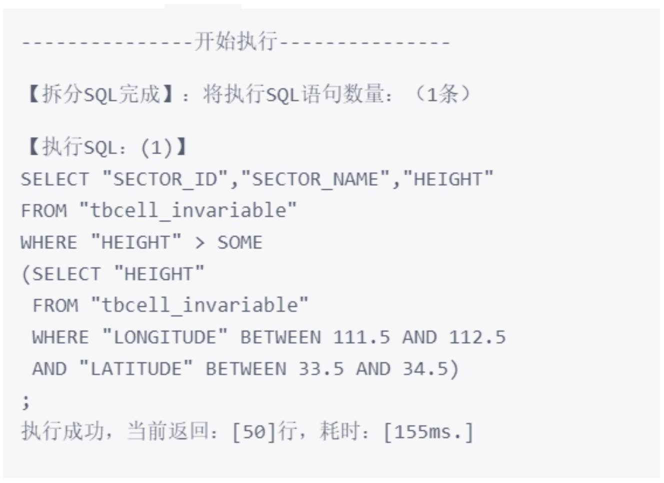


查询 17-2：从一阶邻区表 tbAdjCell、二阶邻区表 tbSecAdjCell 中，使用 Test for Empty Relations 运算符“not exists”，查询满足下列条件的源小区 S_SECTOR_ID：该小区的一阶邻小区包含其全部二阶邻小区，或者该小区的二阶邻小区包含其全部一阶邻小区。

（2）查询语句

```
SELECT DISTINCT "S_SECTOR_ID"
FROM "tbAdjCell_invariable" as A
WHERE NOT EXISTS
(
  (
    SELECT "N_SECTOR_ID"
    FROM "tbSecAdjCell" as B
    WHERE B."S_SECTOR_ID" = A."S_SECTOR_ID"
  )
  EXCEPT
  (
    SELECT "N_SECTOR_ID"
    FROM "tbAdjCell_invariable" as C
    WHERE A."S_SECTOR_ID" = C."S_SECTOR_ID"
  )
)
OR NOT EXISTS
(
  (
    SELECT "N_SECTOR_ID"
    FROM "tbAdjCell_invariable" as D
    WHERE D."S_SECTOR_ID" = A."S_SECTOR_ID"
  )
  EXCEPT
  (
    SELECT "N_SECTOR_ID"
    FROM "tbSecAdjCell" as E
    WHERE A."S_SECTOR_ID" = E."S_SECTOR_ID"
  )
)
;
```

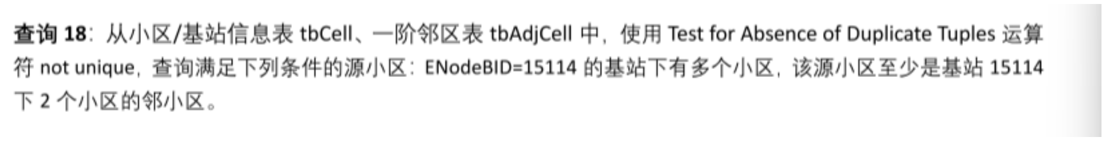
（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT "SECTOR_ID","SECTOR_NAME","HEIGHT"
FROM "tbcell_invariable"
WHERE "HEIGHT" > SOME
(SELECT "HEIGHT"
 FROM "tbcell_invariable"
 WHERE "LONGITUDE" BETWEEN 111.5 AND 112.5
 AND "LATITUDE" BETWEEN 33.5 AND 34.5)
;
执行成功，当前返回：[50]行，耗时：[155ms.]
```

- 查询 18

（1）查询内容



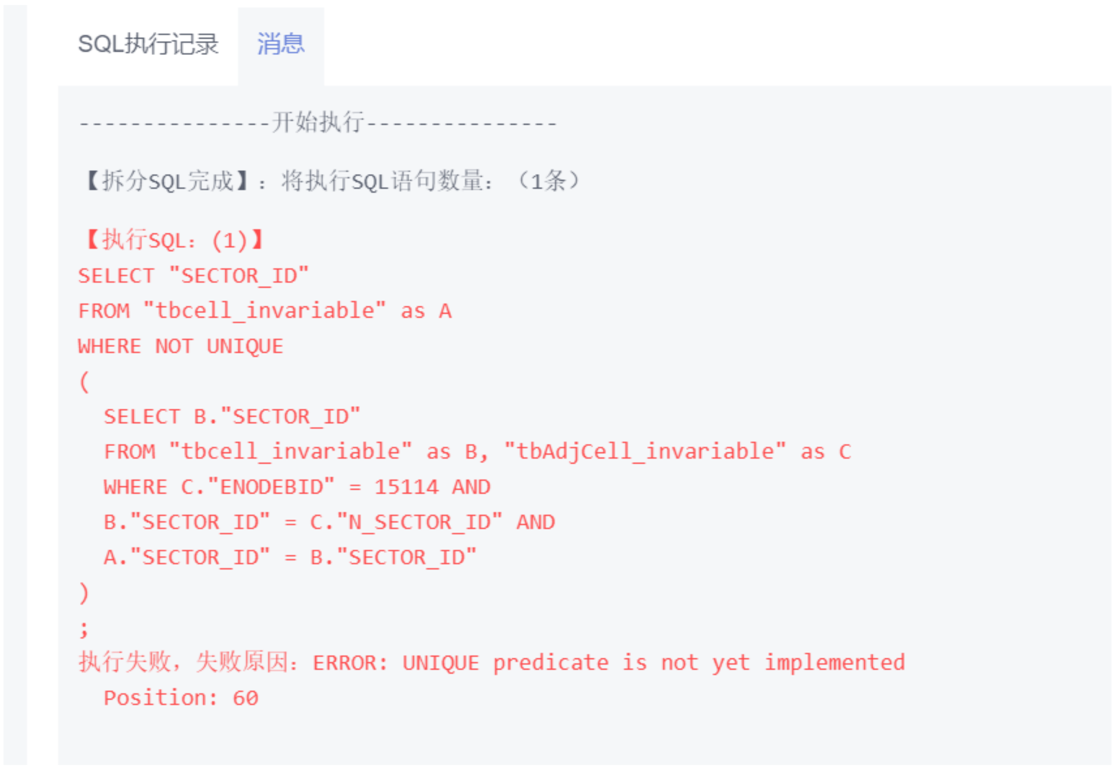
查询 18：从小区/基站信息表 tbCell、一阶邻区表 tbAdjCell 中，使用 Test for Absence of Duplicate Tuples 运算符 not unique，查询满足下列条件的源小区：ENodeBID=15114 的基站下有多个小区，该源小区至少是基站 15114 下 2 个小区的邻小区。

（2）查询语句

```
SELECT "SECTOR_ID"
FROM "tbcell_invariable" as A
WHERE NOT UNIQUE
(
  SELECT B."SECTOR_ID"
  FROM "tbcell_invariable" as B, "tbAdjCell_invariable" as C
  WHERE C."ENODEBID" = 15114 AND
  B."SECTOR_ID" = C."N_SECTOR_ID" AND
  A."SECTOR_ID" = B."SECTOR_ID"
)
;
```

（3）查询结果

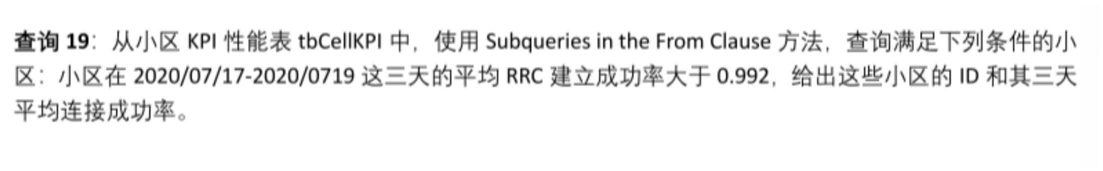
postgre sql 没有 not unique



```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT "SECTOR_ID"
FROM "tbcell_invariable" as A
WHERE NOT UNIQUE
(
  SELECT B."SECTOR_ID"
  FROM "tbcell_invariable" as B, "tbAdjCell_invariable" as C
  WHERE C."ENODEBID" = 15114 AND
  B."SECTOR_ID" = C."N_SECTOR_ID" AND
  A."SECTOR_ID" = B."SECTOR_ID"
)
;
执行失败，失败原因：ERROR: UNIQUE predicate is not yet implemented
  Position: 60
```

- 查询 19

（1）查询内容

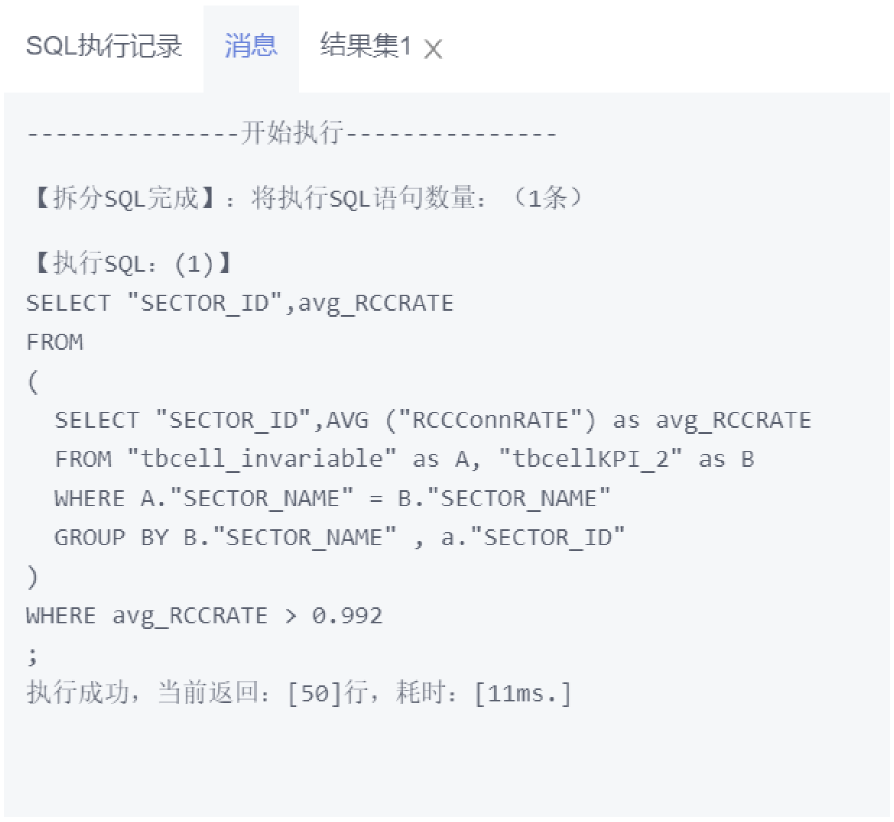


查询 19：从小区 KPI 性能表 tbCellKPI 中，使用 Subqueries in the From Clause 方法，查询满足下列条件的小区：小区在 2020/07/17-2020/0719 这三天的平均 RRC 建立成功率大于 0.992，给出这些小区的 ID 和其三天平均连接成功率。

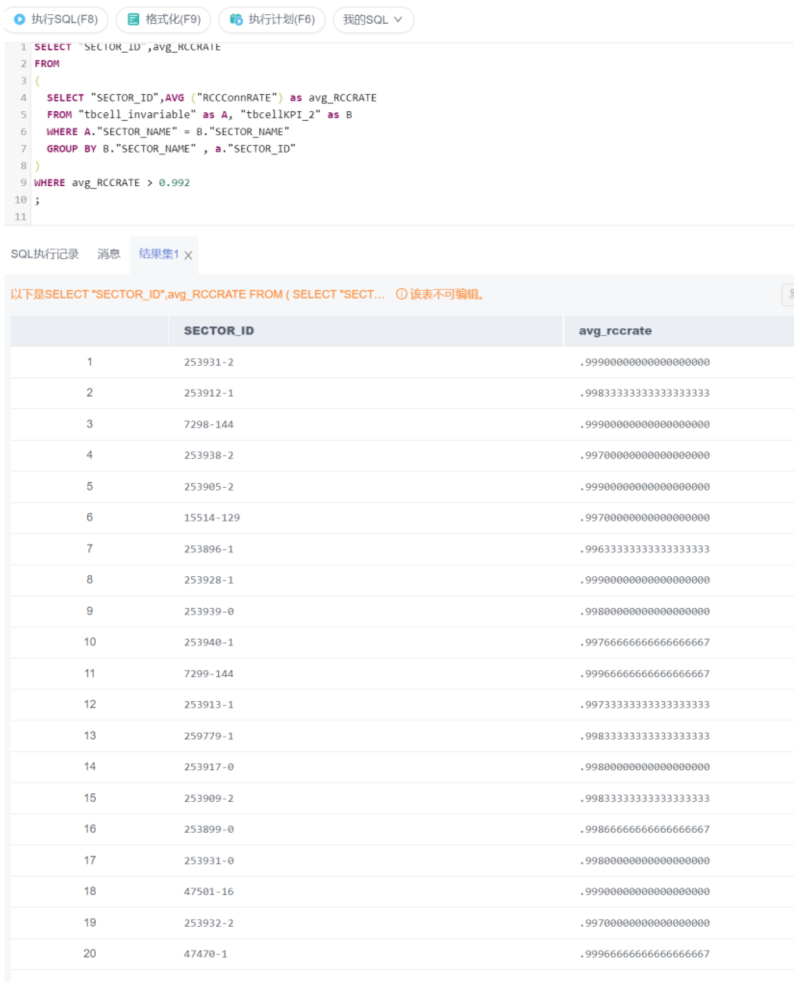
（2）查询语句

```
SELECT "SECTOR_ID",avg_RCCRATE
FROM
(
  SELECT "SECTOR_ID",AVG ("RCCConnRATE") as avg_RCCRATE
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
WHERE avg_RCCRATE > 0.992
;
```

（3）查询结果



```
--------------开始执行--------------

【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】
SELECT "SECTOR_ID",avg_RCCRATE
FROM
(
  SELECT "SECTOR_ID",AVG ("RCCConnRATE") as avg_RCCRATE
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
WHERE avg_RCCRATE > 0.992
;
执行成功, 当前返回: [50]行, 耗时: [11ms.]
```



```
SELECT "SECTOR_ID",avg_RCCRATE
FROM
(
  SELECT "SECTOR_ID",AVG ("RCCConnRATE") as avg_RCCRATE
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
WHERE avg_RCCRATE > 0.992
;
```

以下是SELECT "SECTOR_ID",avg_RCCRATE FROM (SELECT "SECT... ⓘ该表不可编辑。

	SECTOR_ID	avg_rccrate
1	253931-2	.9990000000000000000
2	253912-1	.9983333333333333333
3	7298-144	.9990000000000000000
4	253938-2	.9970000000000000000
5	253905-2	.9990000000000000000
6	15514-129	.9970000000000000000
7	253896-1	.9963333333333333333
8	253928-1	.9990000000000000000
9	253939-0	.9980000000000000000
10	253940-1	.9976666666666666667
11	7299-144	.9996666666666666667
12	253913-1	.9973333333333333333
13	259779-1	.9983333333333333333
14	253917-0	.9980000000000000000
15	253909-2	.9983333333333333333
16	253899-0	.9986666666666666667
17	253931-0	.9980000000000000000
18	47501-16	.9990000000000000000
19	253932-2	.9970000000000000000
20	47470-1	.9996666666666666667

5、 with 临时视图查询

- 查询 20

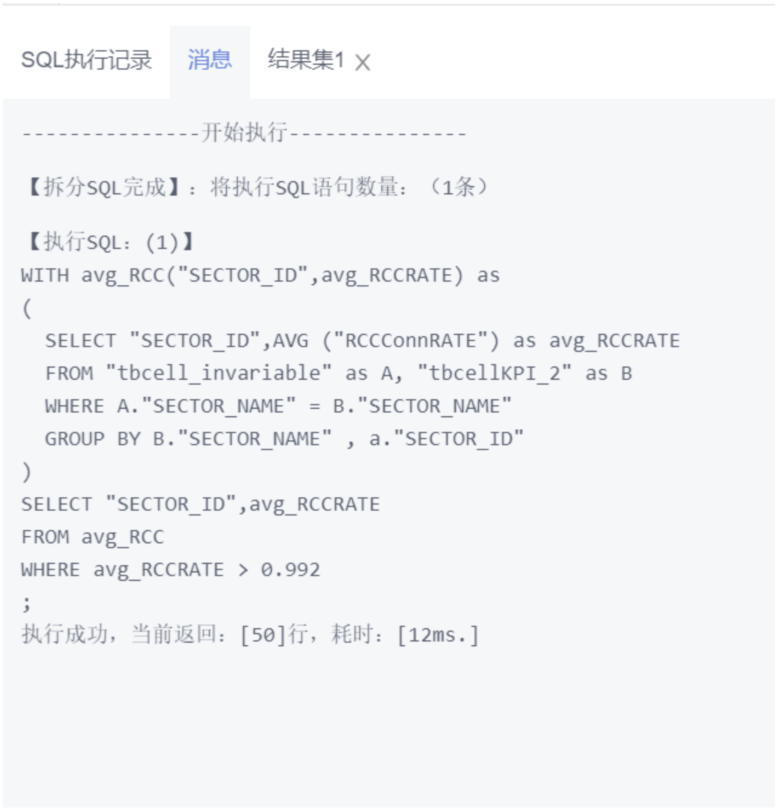
（1）查询内容

查询 20：用 with 临时视图方式，实现查询 19 中查询要求。

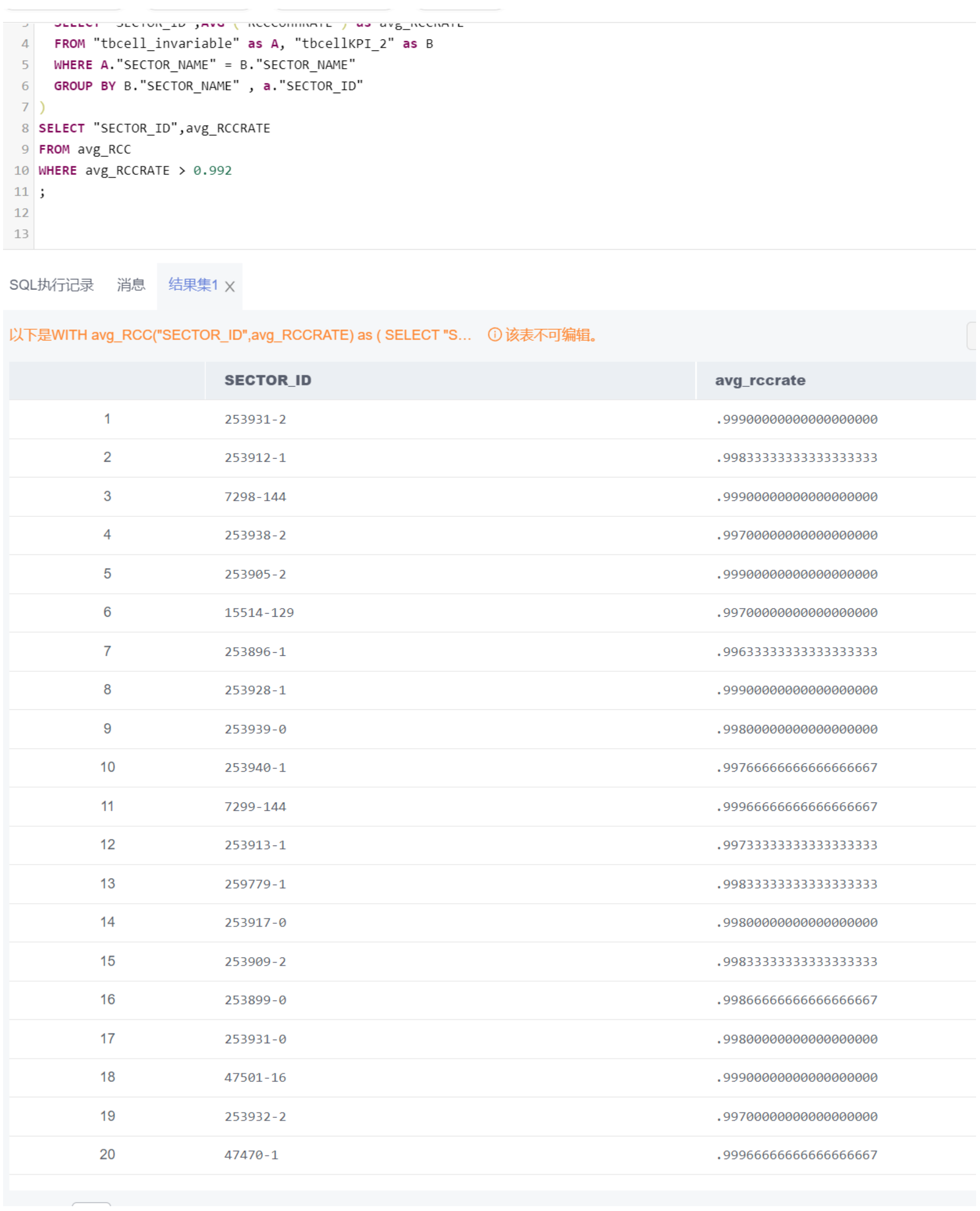
（2）查询语句

```sql
WITH avg_RCC("SECTOR_ID",avg_RCCRATE) as
(
  SELECT "SECTOR_ID",AVG ("RCCConnRATE") as avg_RCCRATE
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
SELECT "SECTOR_ID",avg_RCC
FROM avg_RCCRATE
WHERE avg_RCCRATE.value > 0.992
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】
WITH avg_RCC("SECTOR_ID",avg_RCCRATE) as
(
  SELECT "SECTOR_ID",AVG ("RCCConnRATE") as avg_RCCRATE
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
SELECT "SECTOR_ID",avg_RCCRATE
FROM avg_RCC
WHERE avg_RCCRATE > 0.992
;
执行成功, 当前返回: [50]行, 耗时: [12ms.]
```



```
  FROM "tbcell_invariable" as A, "tbcellKPI_2" as B
  WHERE A."SECTOR_NAME" = B."SECTOR_NAME"
  GROUP BY B."SECTOR_NAME" , a."SECTOR_ID"
)
SELECT "SECTOR_ID",avg_RCCRATE
FROM avg_RCC
WHERE avg_RCCRATE > 0.992
;
```

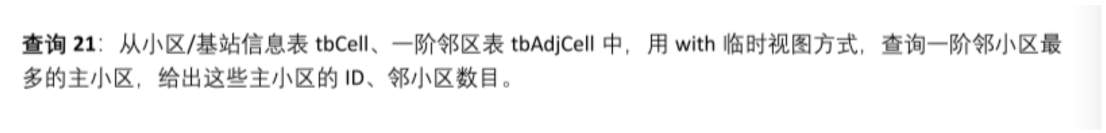
SQL执行记录　消息　结果集1

以下是WITH avg_RCC("SECTOR_ID",avg_RCCRATE) as (SELECT "S...　ⓘ 该表不可编辑。

	SECTOR_ID	avg_rccrate
1	253931-2	.99900000000000000000
2	253912-1	.99833333333333333333
3	7298-144	.99900000000000000000
4	253938-2	.99700000000000000000
5	253905-2	.99900000000000000000
6	15514-129	.99700000000000000000
7	253896-1	.99633333333333333333
8	253928-1	.99900000000000000000
9	253939-0	.99800000000000000000
10	253940-1	.99766666666666666667
11	7299-144	.99966666666666666667
12	253913-1	.99733333333333333333
13	259779-1	.99833333333333333333
14	253917-0	.99800000000000000000
15	253909-2	.99833333333333333333
16	253899-0	.99866666666666666667
17	253931-0	.99800000000000000000
18	47501-16	.99900000000000000000
19	253932-2	.99700000000000000000
20	47470-1	.99966666666666666667

- 查询 21

（1）查询内容

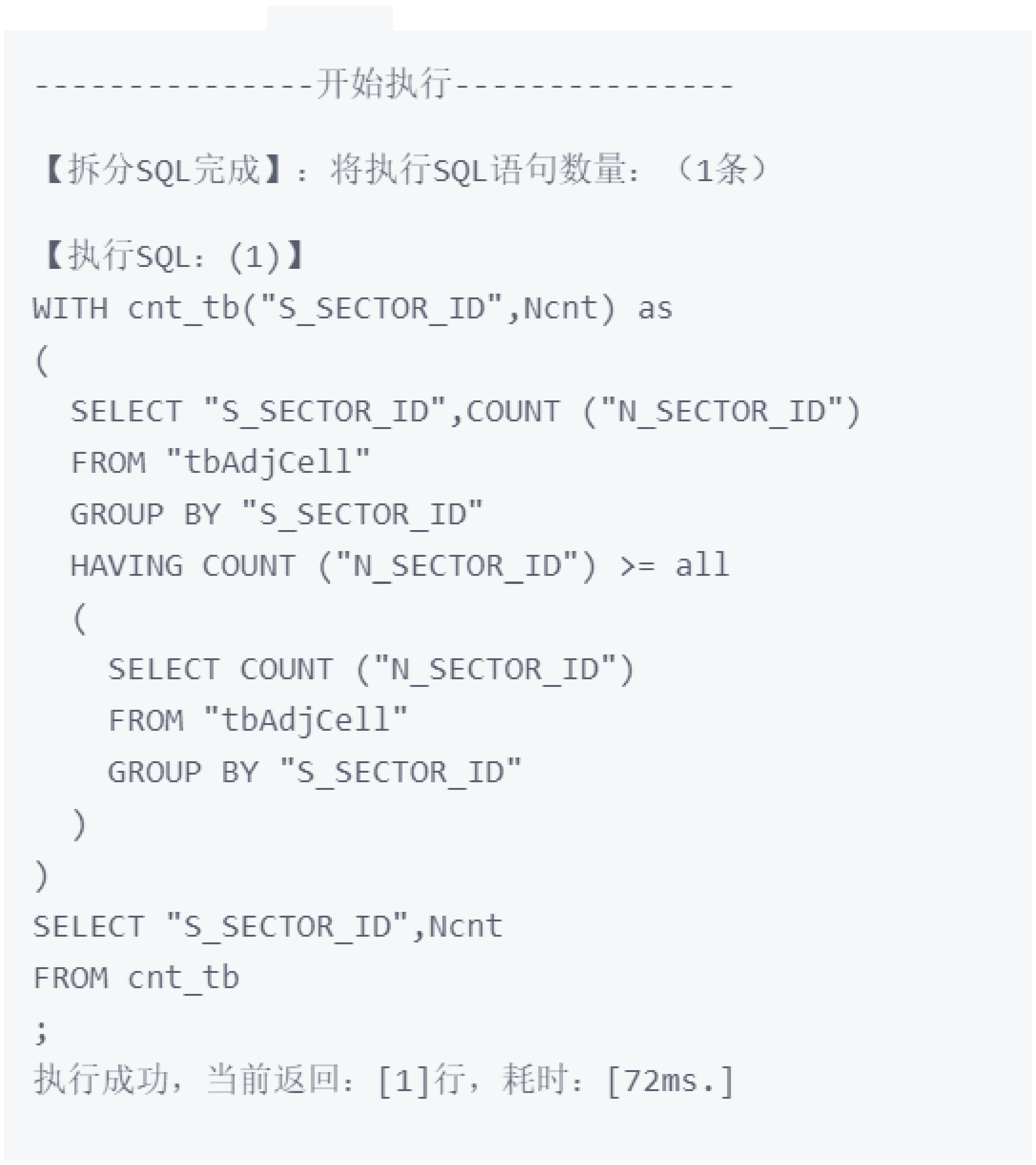


查询 21：从小区/基站信息表 tbCell、一阶邻区表 tbAdjCell 中，用 with 临时视图方式，查询一阶邻小区最多的主小区，给出这些主小区的 ID、邻小区数目。

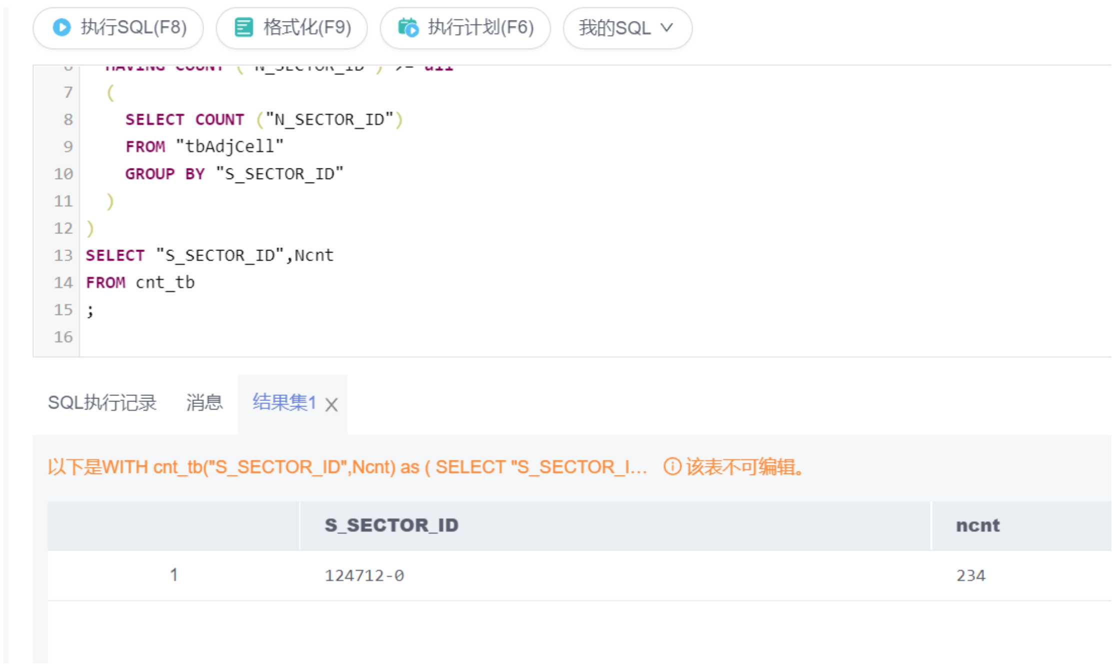
（2）查询语句

```
WITH cnt_tb("S_SECTOR_ID",Ncnt) as
(
  SELECT "S_SECTOR_ID",COUNT ("N_SECTOR_ID")
  FROM "tbAdjCell"
  GROUP BY "S_SECTOR_ID"
  HAVING COUNT ("N_SECTOR_ID") >= all
  (
    SELECT COUNT ("N_SECTOR_ID")
    FROM "tbAdjCell"
    GROUP BY "S_SECTOR_ID"
  )
)
SELECT "S_SECTOR_ID",Ncnt
FROM cnt_tb
;
```

（3）查询结果



```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: （1条）

【执行SQL: (1)】
WITH cnt_tb("S_SECTOR_ID",Ncnt) as
(
  SELECT "S_SECTOR_ID",COUNT ("N_SECTOR_ID")
  FROM "tbAdjCell"
  GROUP BY "S_SECTOR_ID"
  HAVING COUNT ("N_SECTOR_ID") >= all
  (
    SELECT COUNT ("N_SECTOR_ID")
    FROM "tbAdjCell"
    GROUP BY "S_SECTOR_ID"
  )
)
SELECT "S_SECTOR_ID",Ncnt
FROM cnt_tb
;
执行成功, 当前返回: [1]行, 耗时: [72ms.]
```



```sql
  HAVING COUNT ( N_SECTOR_ID ) >= all
  (
    SELECT COUNT ("N_SECTOR_ID")
    FROM "tbAdjCell"
    GROUP BY "S_SECTOR_ID"
  )
)
SELECT "S_SECTOR_ID",Ncnt
FROM cnt_tb
;
```

以下是WITH cnt_tb("S_SECTOR_ID",Ncnt) as (SELECT "S_SECTOR_I... ⓘ 该表不可编辑。

	S_SECTOR_ID	ncnt
1	124712-0	234

6、 键/函数依赖分析

● 查询 22

（1）查询内容

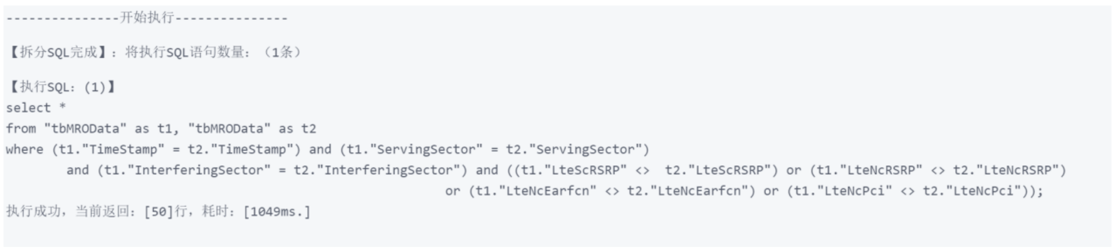
查询 22：在 MRO 测量报告数据表 tbMROData 中，检查 TimeStamp、ServingSector、InterferingSector 是否组成超键。

（2）查询语句

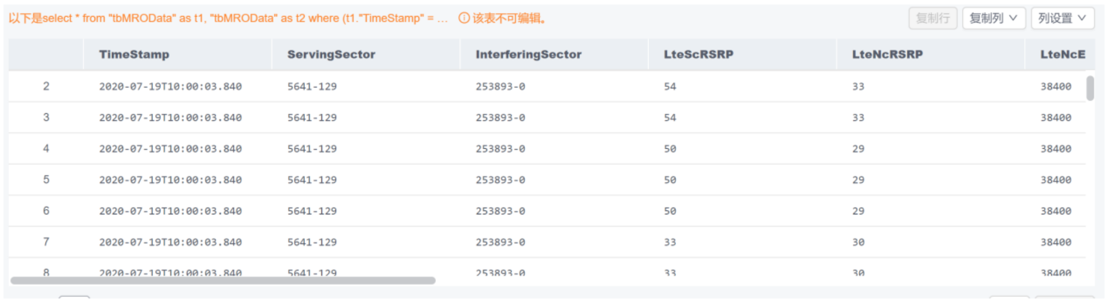
```sql
select *
from "tbMROData" as t1, "tbMROData" as t2
where (t1."TimeStamp" = t2."TimeStamp") and (t1."ServingSector" = t2."ServingSector")
    and (t1."InterferingSector" = t2."InterferingSector") and ((t1."LteScRSRP" <>  t2."LteScRSRP") or (t1."LteNcRSRP" <> t2."LteNcRSRP")
                                                    or (t1."LteNcEarfcn" <> t2."LteNcEarfcn") or (t1."LteNcPci" <> t2."LteNcPci"));
```

（3）查询结果

如图所示，结果元组非空，所以 TimeStamp、ServingSector、InterferingSector 不组成超键。



```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】
select *
from "tbMROData" as t1, "tbMROData" as t2
where (t1."TimeStamp" = t2."TimeStamp") and (t1."ServingSector" = t2."ServingSector")
        and (t1."InterferingSector" = t2."InterferingSector") and ((t1."LteScRSRP" <>  t2."LteScRSRP") or (t1."LteNcRSRP" <> t2."LteNcRSRP")
                                                     or (t1."LteNcEarfcn" <> t2."LteNcEarfcn") or (t1."LteNcPci" <> t2."LteNcPci"));
执行成功, 当前返回: [50]行, 耗时: [1049ms.]
```



以下是select * from "tbMROData" as t1, "tbMROData" as t2 where (t1."TimeStamp" = ... ⓘ该表不可编辑。

	TimeStamp	ServingSector	InterferingSector	LteScRSRP	LteNcRSRP	LteNcE
2	2020-07-19T10:00:03.840	5641-129	253893-0	54	33	38400
3	2020-07-19T10:00:03.840	5641-129	253893-0	54	33	38400
4	2020-07-19T10:00:03.840	5641-129	253893-0	50	29	38400
5	2020-07-19T10:00:03.840	5641-129	253893-0	50	29	38400
6	2020-07-19T10:00:03.840	5641-129	253893-0	50	29	38400
7	2020-07-19T10:00:03.840	5641-129	253893-0	33	30	38400
8	2020-07-19T10:00:03.840	5641-129	253893-0	33	30	38400

● 查询 23

（1）查询内容

查询 23：在 PCI 优化分配表 tbPCIAssignment 中，利用 SQL 语句检查函数依赖 ENODEB_ID→PCI 是否成立；如果不成立，利用 SQL 语句找出导致函数依赖不成立的元组。

（2）查询语句

首先创建 tbPCIAssignment 表：

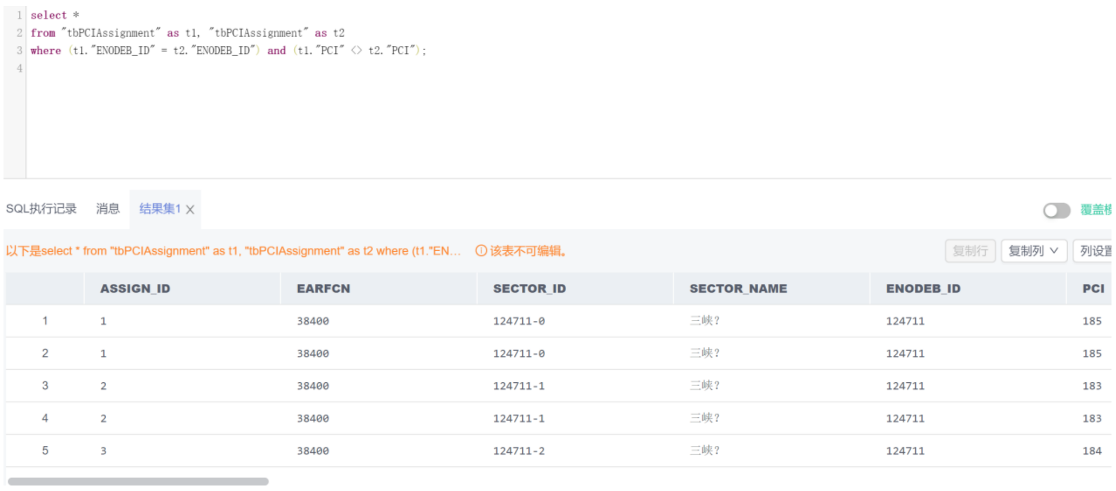
```sql
create table "tbPCIAssignment"
(
  "ASSIGN_ID" serial,
  "EARFCN" int,
  "SECTOR_ID" varchar(50),
  "SECTOR_NAME" varchar(200),
  "ENODEB_ID" int,
  "PCI" int,
  "PSS" int,
  "SSS" int,
  "LONGITUDE" float,
  "LATITUDE" float,
  "STYLE" varchar(50),
  "OPT_DATETIME" timestamp,
  check ("EARFCN" >= 37900),
  primary key ("ASSIGN_ID", "SECTOR_ID")
);
```

寻找函数依赖不成立的元组：

```sql
select *
from "tbPCIAssignment" as t1, "tbPCIAssignment" as t2
where (t1."ENODEB_ID" = t2."ENODEB_ID") and (t1."PCI" <> t2."PCI");
```

（3）查询结果

```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
select *
from "tbPCIAssignment" as t1, "tbPCIAssignment" as t2
where (t1."ENODEB_ID" = t2."ENODEB_ID") and (t1."PCI" <> t2."PCI");
执行成功，当前返回：[50]行，耗时：[24ms.]
```



```
select *
from "tbPCIAssignment" as t1, "tbPCIAssignment" as t2
where (t1."ENODEB_ID" = t2."ENODEB_ID") and (t1."PCI" <> t2."PCI");
```

以下是select * from "tbPCIAssignment" as t1, "tbPCIAssignment" as t2 where (t1."EN… ⓘ 该表不可编辑。

	ASSIGN_ID	EARFCN	SECTOR_ID	SECTOR_NAME	ENODEB_ID	PCI
1	1	38400	124711-0	三峡?	124711	185
2	1	38400	124711-0	三峡?	124711	185
3	2	38400	124711-1	三峡?	124711	183
4	2	38400	124711-1	三峡?	124711	183
5	3	38400	124711-2	三峡?	124711	184

7、 表的插入、删除、更新

● 查询 24

（1）查询内容

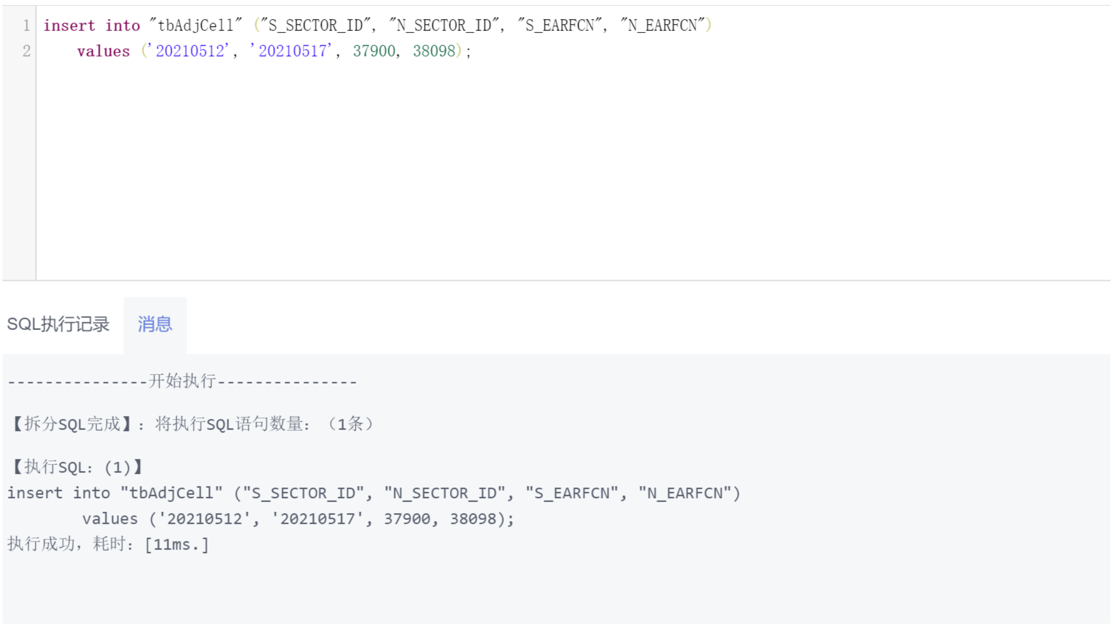
查询 24：向小区一阶邻区关系表中插入一条邻区数据;

（2）查询语句

```
1. insert into "tbAdjCell" ("S_SECTOR_ID", "N_SECTOR_ID", "S_EARFCN", "N_EARFCN
   ")
2.     values ('20210512', '20210517', 37900, 38098);
```

（3）查询结果

如图所示，插入成功。



```
insert into "tbAdjCell" ("S_SECTOR_ID", "N_SECTOR_ID", "S_EARFCN", "N_EARFCN")
    values ('20210512', '20210517', 37900, 38098);
```

SQL执行记录　消息

```
---------------开始执行---------------
【拆分SQL完成】: 将执行SQL语句数量: (1条)
【执行SQL: (1)】
insert into "tbAdjCell" ("S_SECTOR_ID", "N_SECTOR_ID", "S_EARFCN", "N_EARFCN")
       values ('20210512', '20210517', 37900, 38098);
执行成功, 耗时: [11ms.]
```

- 查询 25

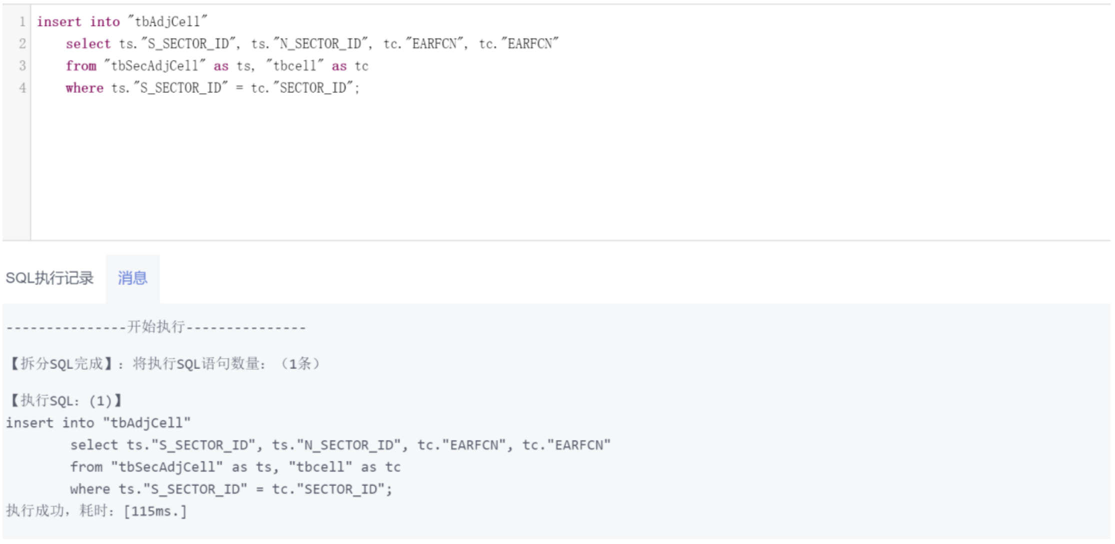
（1）查询内容

查询 25：将小区 124673-0 的全部二阶邻小区，作为该小区的一阶邻小区，加入到一阶邻区表 tbAdjCell 中。

（2）查询语句

```
1. insert into "tbAdjCell"
2.     select ts."S_SECTOR_ID", ts."N_SECTOR_ID", tc."EARFCN", tc."EARFCN"
3.     from "tbSecAdjCell" as ts, "tbcell" as tc
4.     where ts."S_SECTOR_ID" = tc."SECTOR_ID";
```

（3）查询结果



```
insert into "tbAdjCell"
    select ts."S_SECTOR_ID", ts."N_SECTOR_ID", tc."EARFCN", tc."EARFCN"
    from "tbSecAdjCell" as ts, "tbcell" as tc
    where ts."S_SECTOR_ID" = tc."SECTOR_ID";
```

SQL执行记录　消息

```
---------------开始执行---------------
【拆分SQL完成】: 将执行SQL语句数量: (1条)
【执行SQL: (1)】
insert into "tbAdjCell"
       select ts."S_SECTOR_ID", ts."N_SECTOR_ID", tc."EARFCN", tc."EARFCN"
       from "tbSecAdjCell" as ts, "tbcell" as tc
       where ts."S_SECTOR_ID" = tc."SECTOR_ID";
执行成功, 耗时: [115ms.]
```

- 查询 26

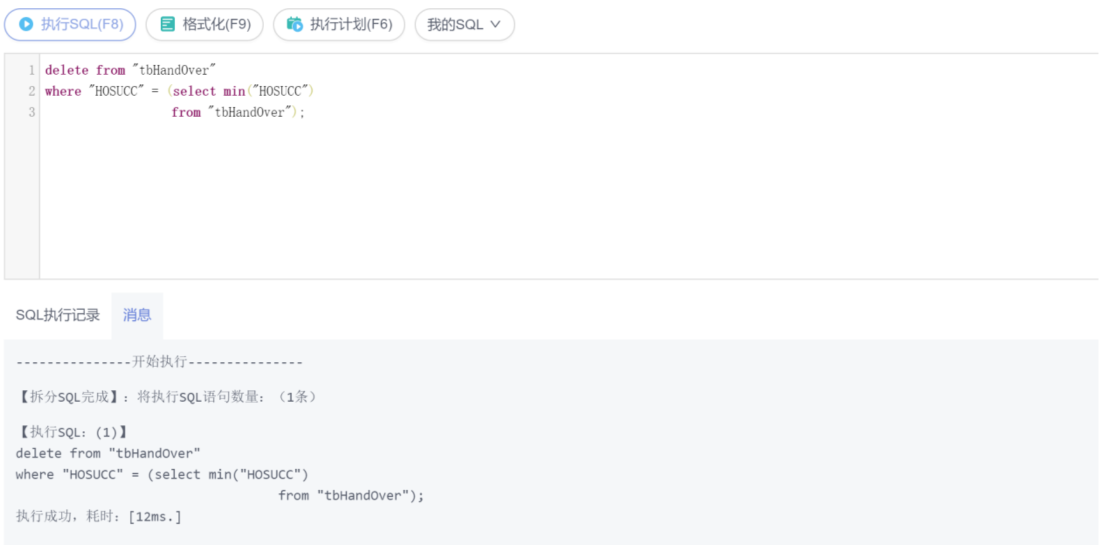
（1）查询内容

查询 26：在小区切换统计性能表 tbHandover 中，删除切换次数最少的那些切换数据。

（2）查询语句

```
delete from "tbHandOver"
where "HOSUCC" = (select min("HOSUCC")
                  from "tbHandOver");
```

（3）查询结果



```
delete from "tbHandOver"
where "HOSUCC" = (select min("HOSUCC")
                  from "tbHandOver");
```

```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
delete from "tbHandOver"
where "HOSUCC" = (select min("HOSUCC")
                  from "tbHandOver");
执行成功，耗时：[12ms.]
```

- 查询 27

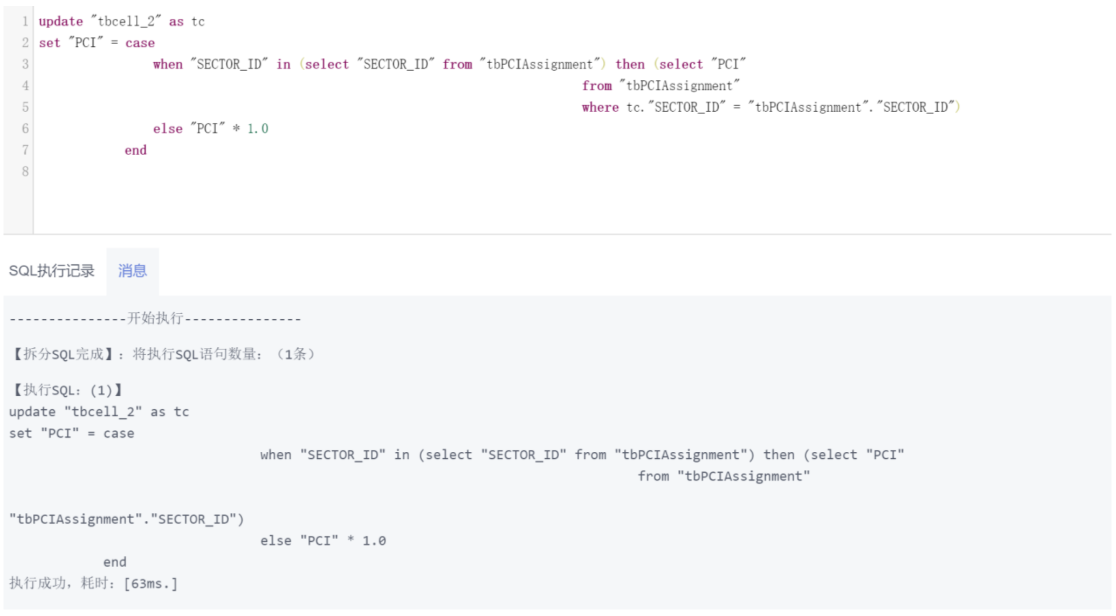
（1）查询内容

查询27：用小区PCI优化调整结果表tbPCIAssignmeng给出的各个小区的优化调整后的PCI值，替换小区/基站消息表tbCell中对应小区的物理小区标识PCI。

（2）查询语句

```
update "tbcell_2" as tc
set "PCI" = case
                when "SECTOR_ID" in (select "SECTOR_ID"
                                     from "tbPCIAssignment")
                                     then (select "PCI"
                                        from "tbPCIAssignment"
                                        where tc."SECTOR_ID" = "tbPCIAssignment"."SECTOR_ID")
                else "PCI" * 1.0
            end
```

（3）查询结果



```
update "tbcell_2" as tc
set "PCI" = case
                when "SECTOR_ID" in (select "SECTOR_ID" from "tbPCIAssignment") then (select "PCI"
                                                                                        from "tbPCIAssignment"
                                                                                        where tc."SECTOR_ID" = "tbPCIAssignment"."SECTOR_ID")
                else "PCI" * 1.0
            end
```

```
---------------开始执行---------------

【拆分SQL完成】: 将执行SQL语句数量: (1条)

【执行SQL: (1)】
update "tbcell_2" as tc
set "PCI" = case
                                when "SECTOR_ID" in (select "SECTOR_ID" from "tbPCIAssignment") then (select "PCI"
                                                                                                   from "tbPCIAssignment"

"tbPCIAssignment"."SECTOR_ID")
                                else "PCI" * 1.0
            end
执行成功, 耗时: [63ms.]
```

- 查询 28

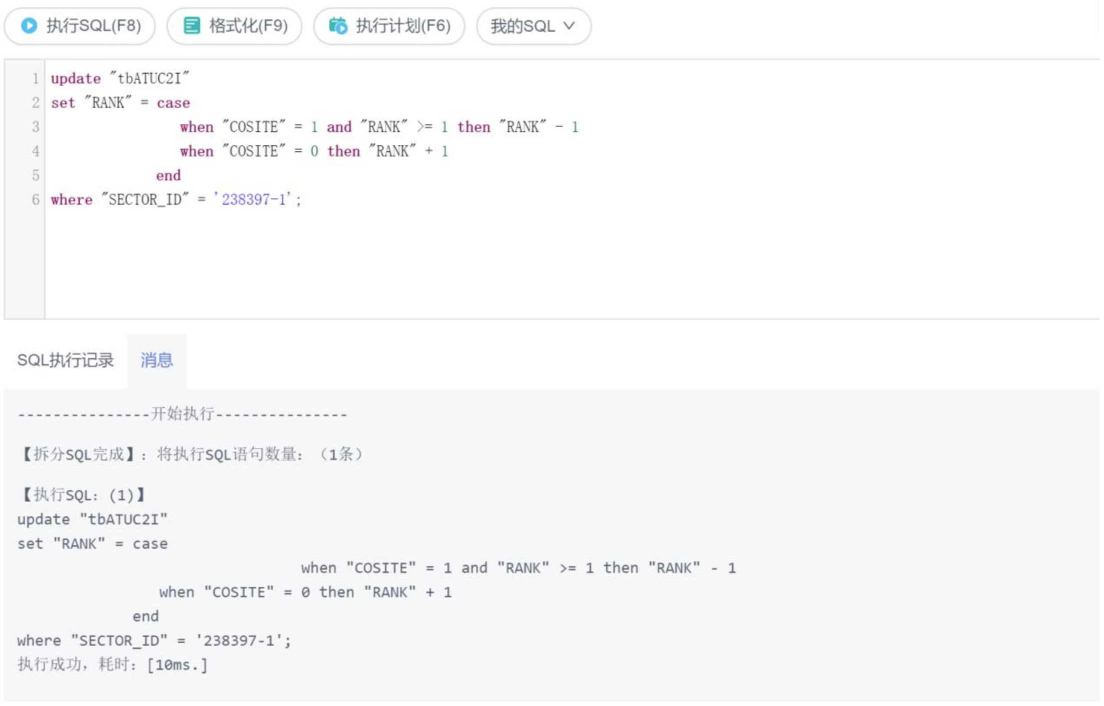
(1) 查询内容

查询 28: 针对路测 ATU 干扰矩阵表 tbATUC2I, 使用 update/case 语句做出如下修改: 对主小区 SECTOR_ID=238397-1, 如果该小区与干扰小区 N_SECTOR_ID 为同站小区(cosite=1)且干扰强度排序 rank 不小于 1, 则干扰强度排序减 1; 如果主小区与干扰小区不同站, 干扰强度排序加 1。

(2) 查询语句

```
update "tbATUC2I"
set "RANK" = case
                when "COSITE" = 1 and "RANK" >= 1 then "RANK" - 1
                when "COSITE" = 0 then "RANK" + 1
             end
where "SECTOR_ID" = '238397-1';
```

(3) 查询结果



```
update "tbATUC2I"
set "RANK" = case
                when "COSITE" = 1 and "RANK" >= 1 then "RANK" - 1
                when "COSITE" = 0 then "RANK" + 1
             end
where "SECTOR_ID" = '238397-1';
```

```
---------------开始执行---------------
【拆分SQL完成】: 将执行SQL语句数量: (1条)
【执行SQL: (1)】
update "tbATUC2I"
set "RANK" = case
                                when "COSITE" = 1 and "RANK" >= 1 then "RANK" - 1
                when "COSITE" = 0 then "RANK" + 1
             end
where "SECTOR_ID" = '238397-1';
执行成功, 耗时: [10ms.]
```

三、遇到问题及解决

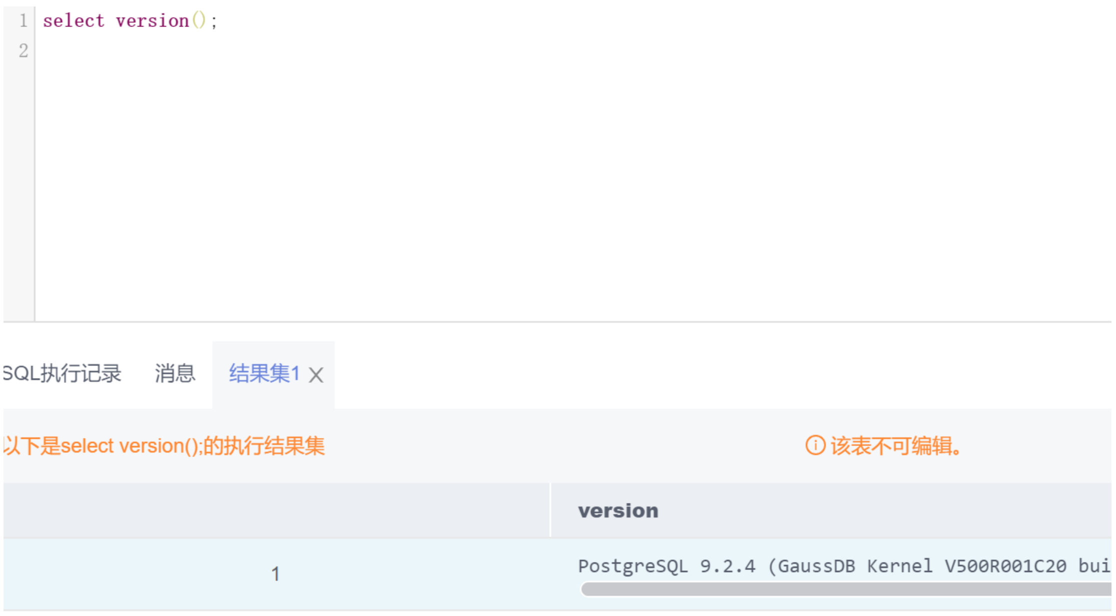
1、版本问题

在建立 tbPCIAssignment 表的过程中，设置 ASSIGN_ID 的自增首先采用 IDENTITY ，发现无法编译通过。查阅了 Stack Overflow 后发现，在 PostgreSQL 10 之前，自增只能使用 serial 或者 bigserial。

The SQL standard `IDENTITY` was added in PostgreSQL 10 but your server (which does all the real work) is 9.4. Before 10 you have to use `serial` or `bigserial` types:

```
CREATE TABLE distributors (
     did    serial not null primary key,
     name   varchar(40) NOT NULL CHECK (name <> '')
);
```

使用 select version() 后发现,当前数据库的 PostgreSQL 版本为 9.2.4,低于 10,改用 serial，问题得以解决。


```
select version();
```

以下是select version();的执行结果集　　　　ⓘ 该表不可编辑。

	version
1	PostgreSQL 9.2.4 (GaussDB Kernel V500R001C20 bui

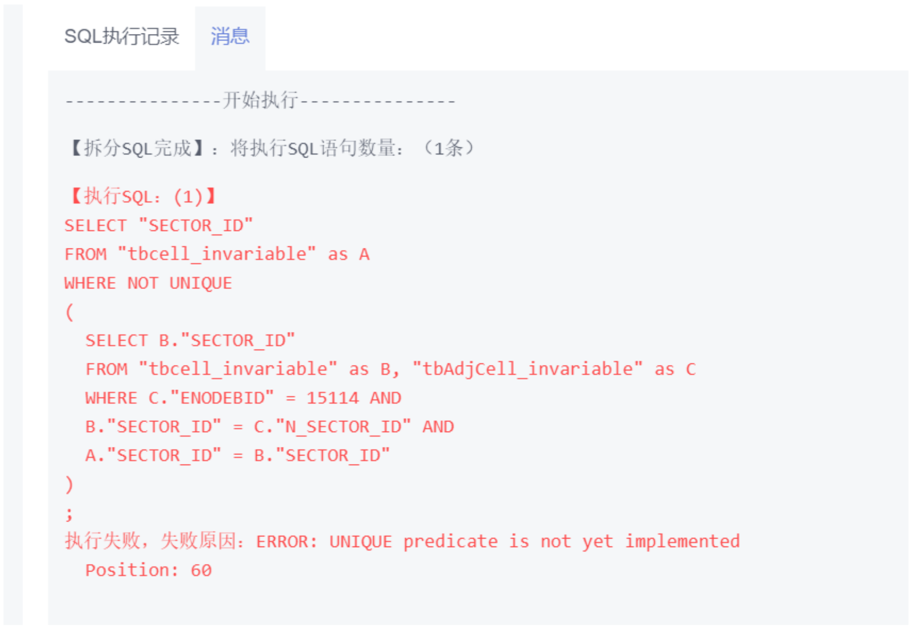
2、delete 测试

对于查询 26，我首先考虑删除一个元组后，“=”后的 select 语句可能会发生改变，后来查阅了教科书，发现 delete 在执行删除操作之前，会进行所有元组的测试，这是至关重要的，问题得以解决。

```
1. delete from "tbHandOver"
2. where "HOSUCC" = (select min("HOSUCC")
3.                   from "tbHandOver");
```

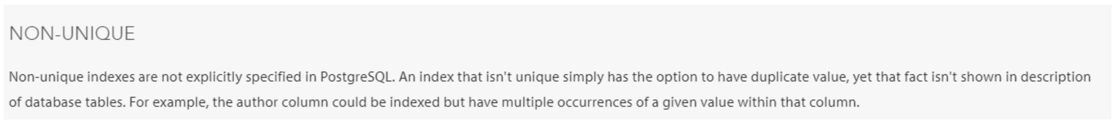
3、not unique 问题

对于查询 18，运行 not unique：



```
---------------开始执行---------------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
SELECT "SECTOR_ID"
FROM "tbcell_invariable" as A
WHERE NOT UNIQUE
(
  SELECT B."SECTOR_ID"
  FROM "tbcell_invariable" as B, "tbAdjCell_invariable" as C
  WHERE C."ENODEBID" = 15114 AND
  B."SECTOR_ID" = C."N_SECTOR_ID" AND
  A."SECTOR_ID" = B."SECTOR_ID"
)
;
执行失败，失败原因：ERROR: UNIQUE predicate is not yet implemented
  Position: 60
```

经过对 postgre sql 文档、stackoverflow 的查询，我们发现 not unique 并没有在 postgre sql 中显式定义：



NON-UNIQUE

Non-unique indexes are not explicitly specified in PostgreSQL. An index that isn't unique simply has the option to have duplicate value, yet that fact isn't shown in description of database tables. For example, the author column could be indexed but have multiple occurrences of a given value within that column.