## 3.2 信息查询

至少完成以下几类信息的查询

1. 小区配置信息查询

从tbCell表中查询小区信息。

要求：

1. 用户在界面上输入小区ID Cell\_ID/Sector\_ID或小区名称CellName/SectorName，以列表方式输出该小区全部信息；

根据题目要求，对数据库进行查询操作时，有两个依据标准，一是小区ID CellID/Sector\_ID，二是小区名称CellName/SectorName，因此可以针对这两个属性，在tbCell表上建立索引。按照习惯，我们选择在CellID/Sector\_ID属性上建立主键，同时也就生成了聚集索引，那么，在CellName/SectorName属性上建立非聚集索引。

由于聚集索引和非聚集索引自身的原因，我们在针对CellID/Sector\_ID查询时，既可以判断相等(即where CellID/Sector\_ID = $CellID/$Sector\_ID)，也可以判断不等（即where CellID/Sector\_ID > $CellID/$Sector\_ID 或者 where CellID/Sector\_ID < $CellID/$Sector\_ID）。在针对CellName/SectorName查询时，只能判断是否相等（即where CellName/SectorName = $CellName/$Sector\_Name）。

在tbCell表的CellID/Sector\_ID属性上建立主键的语句如下，同时自动生成一个聚集索引

alter table tbCell add constraint pk\_sector\_id primary key (SECTOR\_ID)

在tbCell表的CellName/SectorName属性上建立非聚集索引的语句如下：

create nonclustered index SNameIndex on tbCell(SECTOR\_NAME)

完成上述工作后，只要用户明确CellID/Sector\_ID或CellName/SectorName，就可以调用查询语句获得小区信息。

通过CellID/Sector\_ID查询的语句如下：

select \* from tbCell where SECTOR\_ID = $SECTOR\_ID

通过CellName/SectorName查询的语句如下：

select \* from tbCell where SECTOR\_NAME = $SECTOR\_NAME

用户输入CellID/Sector\_ID或CellName/SectorName的方式，扩展性非常差，也就是说，只有在用户非常清晰地了解tbCell表的结构，并且熟识这些属性下有哪些值才能完成查询操作，一旦记忆偏差，或者表的结构发生改变，或者增加减少了部分数据，都会导致这次查询的失败，或者无法充分地利用到整个数据表。因此，这种方法只是试验阶段的一种方便debug的手段，在正式版本中，是不应该出现 的。

1. 程序在查询界面下来列表中程序tbCell中全部小区名称，用户在界面上通过下拉列表选择特定小区名称，以列表方式输出该小区全部信息；

相比于（1），此问题描述要求展示全部小区名称，增加了扩展性和可用性。需要注意的是，展示的是全部小区名称，而不是全部小区ID，原因在于，小区ID是由数字组成的字符串，向用户展示冗长的数字串，是不友好的，更何况是一个巨大的数据表中的冗长数字串，处于这个考虑，只向用户展示全部小区名称，小区名称同样也能唯一确定数据行。

要完成这个要求，需要在（1）的基础上，增加查询全部小区名的操作，语句如下：

select SECTOR\_NAME from tbCell

根据查询到的SectorName更新图形化界面的下拉列表，保证了数据的有效性，可扩展性，即使是对表增删操作，也不影响用户的使用。

1. 基站eNodeB信息查询

从tbCell表中查询基站所属全部小区信息。

要求：

用户在界面上输入、或者以下拉列表方式选择基站ID eNodeBID或基站名称eNodeBName，以列表方式输出该基站全部小区信息；

首先讨论用户在界面上输入eNodeBID或eNodeBName的方式，该方式同1（1），仅适用于对数据系统非常了解，并实时跟进数据的用户，一般不存在这种用户，因此，这种方式也就适用于debug阶段。

该方式下的查询语句如下：

根据ENodeBID查询

select \* from tbCell where eNODEBID = $ENODEBID

根据EnodeBName查询

select \* from tbCell where eNODEB\_NAME = $ENODEB\_NAME

然后讨论下拉列表选择的方式，该方式优势同1（2）对1（1）的优势，扩展性高，可用性强。但是要完成这个要求，同样需要在用户选择时实时更新下拉列表的数据，这就需要对eNodeBID和eNodeBName查询，查询语句如下：

查询eNodeBID:

select distinct ENODEBID from tbCell order by ENODEBID asc

查询eNodeBName：

select distinct ENODEB\_NAME from tbCell order by ENODEB\_NAME asc

由于eNodeBID和eNodeBName不唯一，也就是有多个小区处于同一个eNodeB，因此，在查询时只需要查询出单一的eNodeBID或eNodeName即可，这也就是distinct的用武之处，对查询到的eNodeBID和eNodeBName按字母序升序排列，方便用户查找想要的eNodeBID或eNodeBName。至此，完成（2）的要求。

1. KPI指标信息查询

通过人机界面对话框选定表tbKPI中的某个特定网元，查询数据库，列出该网元从2016年7月17日到7月19日的RRC连接重建比率 (%)，并将结果用柱状图或折线图的方式呈现出来，要求查询至少3个网元。

问题描述可以分为两个具体步骤：step1，查询所有网元名称并向用户展示，step2，根据用户所选网元名称查询该网元从2016年7月17日到7月19日的RRC连接重建比率 (%)，并绘制统计图。

step1:查询所有网元名称，由于网元名称在此表中不是唯一的存在，因此在查询时只需要查找单一的网元名称即可，同时我们也将其按照字母序升序排列，方便用户检索。

上述操作的查询语句如下：

select distinct 网元名称 from tbKPI order by 网元名称 asc

step2：根据用户选定的网元名称，查询对应的起始时间，小区1，RRC连接重建比率 (%)。之所以要将小区1查询出，是出于同一个网元名称，对应多个不同的小区1的考虑，因此需要将其全部列出，分别绘制统计图。

上述操作的查询语句如下：

select 起始时间 , 小区1 , [RRC连接重建比率 (%)] from tbKPI where 网元名称 = $网元名称order by 起始时间 asc, 小区1 asc

首先按照起始时间升序排列，然后按照小区1升序排列，查询结果示例如下：

起始时间 小区1 RRC连接重建比率 (%)

07/17/2016 00:00:00 三门峡义马常村-HLHF-1 0.006

07/17/2016 00:00:00 三门峡义马常村-HLHF-2 0.005

07/17/2016 00:00:00 三门峡义马常村-HLHF-3 0.013

07/18/2016 00:00:00 三门峡义马常村-HLHF-1 0.005

07/18/2016 00:00:00 三门峡义马常村-HLHF-2 0.004

07/18/2016 00:00:00 三门峡义马常村-HLHF-3 0.007

07/19/2016 00:00:00 三门峡义马常村-HLHF-1 0.005

07/19/2016 00:00:00 三门峡义马常村-HLHF-2 0.003

07/19/2016 00:00:00 三门峡义马常村-HLHF-3 0.012

我们可以看到，该示例中，同一网元名称，对应三个小区1.

1. PRB信息统计与查询

（1）根据表“优化区17日-19日每PRB干扰 查询-15分钟”，统计小时级的PRB干扰数据，生成一张新表tbPRBnew，并导出到外部的excel表中。

首先创建tbPRBnew表，表中包含网元名称，日（17-19日的哪一天），时（0-23的某个小时），100个PRB上检测到的干扰噪声平均值（0，1，2…99）

首先创建tbPRBnew表

create table tbPRBnew (

网元名称 nvarchar(50),

日 integer,

时 integer,

[第0个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第1个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第2个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第3个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第4个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第5个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第6个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第7个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第8个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第9个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第10个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第11个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第12个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第13个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第14个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第15个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第16个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第17个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第18个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第19个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第20个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第21个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第22个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第23个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第24个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第25个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第26个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第27个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第28个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第29个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第30个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第31个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第32个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第33个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第34个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第35个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第36个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第37个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第38个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第39个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第40个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第41个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第42个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第43个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第44个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第45个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第46个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第47个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第48个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第49个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第50个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第51个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第52个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第53个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第54个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第55个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第56个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第57个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第58个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第59个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第60个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第61个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第62个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第63个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第64个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第65个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第66个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第67个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第68个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第69个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第70个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第71个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第72个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第73个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第74个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第75个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第76个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第77个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第78个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第79个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第80个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第81个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第82个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第83个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第84个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第85个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第86个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第87个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第88个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第89个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第90个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第91个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第92个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第93个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第94个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第95个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第96个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第97个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第98个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float,

[第99个PRB上检测到的干扰噪声的平均值(毫瓦分贝)] float

)

注：某一网元某一小时内的PRB干扰=该小时内4个15分钟平均值的和/4，要求计算所有的网元从2016年7月17日到7月19日的每个小时的100个PRB上检测到的干扰噪声的平均值。

在计算某一网元某一小时内的PRB干扰=该小时内4个15分钟平均值的和/4时，我们做出这样的考虑，网元的数量巨大，在对每个网元操作时，都有100个PRB属性需要计算平均值，这样算下来，计算次数很大，每次都执行一次这样的语句，SQL编译阶段就会耗费可观的时间，因此，我们考虑将这个操作作为存储过程提前创建，每次需要向tbPRBnew插入时，就调用一次这个存储过程。存储过程与传统SQL语句相比，不在执行时编译，而是提前创建好，创建完成就编译好SQL语句，这样，就免去了执行前进行的SQL编译时间，在反复调用某个查询时，通过存储过程，能省去不少编译时间。

在这里，我们创建存储过程的语句如下：

create procedure PRB\_Hour\_Pro

as

begin

insert into tbPRBnew select 网元名称,DATEPART(dd,起始时间) as 日, DATEPART(hh,起始时间) as 时, avg([第0个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第0个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第1个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第1个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第2个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第2个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第3个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第3个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第4个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第4个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第5个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第5个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第6个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第6个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第7个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第7个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第8个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第8个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第9个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第9个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第10个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第10个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第11个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第11个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第12个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第12个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第13个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第13个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第14个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第14个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第15个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第15个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第16个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第16个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第17个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第17个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第18个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第18个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第19个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第19个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第20个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第20个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第21个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第21个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第22个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第22个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第23个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第23个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第24个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第24个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第25个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第25个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第26个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第26个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第27个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第27个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第28个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第28个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第29个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第29个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第30个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第30个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第31个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第31个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第32个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第32个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第33个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第33个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第34个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第34个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第35个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第35个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第36个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第36个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第37个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第37个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第38个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第38个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第39个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第39个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第40个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第40个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第41个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第41个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第42个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第42个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第43个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第43个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第44个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第44个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第45个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第45个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第46个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第46个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第47个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第47个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第48个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第48个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第49个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第49个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第50个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第50个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第51个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第51个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第52个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第52个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第53个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第53个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第54个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第54个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第55个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第55个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第56个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第56个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第57个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第57个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第58个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第58个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第59个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第59个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第60个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第60个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第61个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第61个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第62个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第62个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第63个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第63个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第64个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第64个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第65个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第65个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第66个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第66个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第67个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第67个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第68个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第68个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第69个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第69个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第70个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第70个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第71个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第71个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第72个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第72个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第73个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第73个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第74个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第74个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第75个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第75个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第76个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第76个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第77个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第77个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第78个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第78个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第79个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第79个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第80个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第80个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第81个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第81个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第82个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第82个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第83个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第83个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第84个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第84个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第85个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第85个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第86个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第86个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第87个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第87个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第88个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第88个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第89个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第89个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第90个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第90个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第91个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第91个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第92个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第92个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第93个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第93个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第94个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第94个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第95个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第95个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第96个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第96个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第97个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第97个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第98个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第98个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] ,avg([第99个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)]) as [第99个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] from tbPRB group by 网元名称, DATEPART(dd,起始时间), DATEPART(hh,起始时间) order by 网元名称 asc, DATEPART(dd,起始时间) asc, DATEPART(hh,起始时间) asc

end

我们可以看到，100多条属性的SQL语句，每次都书写或者拷贝是不现实，且不直观的，万一发生手误的情况，就要涉及对数据版本的回溯，而我们知道，数据操作具有持久性，所以我们不能寄希望于恢复系统或者回滚，只能人工进行逆操作，其工作量和难度，可想而知。因此存储过程是一个上上之法。我们只需要在需要执行上述语句时，执行下述调用存储过程：

exec dbo.PRB\_Hour\_pro

便能完成冗长的语句

查询示例：select \* from tbPRBnew

网元名称 日 时 第0个PRB上检测到的干扰噪声的平均值 (毫瓦分贝) 第1个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)

三门峡义马汽车站-HLHF 17 0 -103.416666666667 -114.666666666667

三门峡义马汽车站-HLHF 17 1 -103.25 -115.333333333333

三门峡义马汽车站-HLHF 17 2 -104.916666666667 -116.083333333333

三门峡义马汽车站-HLHF 17 3 -105 -117.166666666667

三门峡义马汽车站-HLHF 17 4 -106.166666666667 -117

三门峡义马汽车站-HLHF 17 5 -105.166666666667 -117.25

三门峡义马汽车站-HLHF 17 6 -103.833333333333 -114.916666666667

三门峡义马汽车站-HLHF 17 7 -104.416666666667 -114.583333333333

三门峡义马汽车站-HLHF 17 8 -104.75 -113.666666666667

（2）根据（1）中得到的小时级的PRB干扰数据表，通过人机界面对话框选定某个特定网元，查询数据库，列出其某一天从00:00:00到24：00：00每个小时内第60个PRB上检测到的干扰噪声的平均值，并将结果用柱状图或折线图的方式呈现出来。

有了（1）得到的tbPRBnew表，（2）的操作只是简单地查询某行而已

查询语句如下：

select 网元名称, 日, 时, [第60个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] as [第60个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)] from tbPRBnew where 网元名称 = $网元名称

查询示例如下：

网元名称 日 时 第60个PRB上检测到的干扰噪声的平均值 (毫瓦分贝)

三门峡义马汽车站-HLHF 17 0 -116.416666666667

三门峡义马汽车站-HLHF 17 1 -116.583333333333

三门峡义马汽车站-HLHF 17 2 -116.833333333333

三门峡义马汽车站-HLHF 17 3 -116.75

三门峡义马汽车站-HLHF 17 4 -117

三门峡义马汽车站-HLHF 17 5 -117

三门峡义马汽车站-HLHF 17 6 -116.666666666667

## 3.3 主邻小区C2I干扰分析

参照基本课程实验中的tbMROData和tbC2I表。

tbMROData记录了LTE MRO测量报告的内容。在tbMROData中，每一行表示位于服务/主小区ServingSector覆盖范围内特定位置的1个移动终端在一个特定时刻接收到的：1）来自服务/主小区ServingSector参考信号接收功率LteScRSRP；2）来自周边某个邻小区InterfereringSector的参考信号接收功率LteNcRSRP，该信号形成对主小区信号的干扰。

服务/主、干扰/邻小区间的C2I定义为：

C2I(ServingSector, InterfereringSector)= LteScRSRP-LteNcRSRP

C2I值越小，说明邻小区对主小区的干扰越大。

对于一对主邻小区对<ServeringSector, InterfereringSector>，在主小区覆盖范围内，会有许多终端在不同位置处、不同时刻收到许多条MRO测量报告。因此在tbMROData中，一对主邻小区会有许多条<LteScRSRP, LteNcRSRP>RSRP测量值对。

例如<主小区ID=‘5641-129’，邻小区ID=‘15513-128’>有6515条测量值对。

将主邻小区对<ServeringSector, InterfereringSector>间的C2I看作符合正态分布的随机变量，根据tbMROData表中的多条<LteScRSRP, LteNcRSRP>RSRP测量值对，可以计算C2I的均值C2I\_mean和标准差std；

说明：在tbC2I表中，主邻小区分别用SCELL、NCELL表示，有别于tbMROData表。

在此基础上，进一步计算：

1. 主小区信号比邻小区RSRP信号强度弱9db的概率，即主邻小区RSRP差值小于9的概率，

PrbC2I9(SCell, NCell)=Prb{C2I(SCell, NCell)<9}

**PrbC2I9越大，说明邻小区对主小区的干扰越大。**

1. 主小区、邻小区的RSRP信号强度差值小于6db的概率PrbABS6，即

PrbABS6(SCell, NCell)=Prb{ -6<C2I(SCell,NCell)<6}

**PrbABS6越大，说明主邻小区信号相差不大，主邻小区间出现重叠覆盖干扰。**

具体要求和实现步骤如下：

1．主邻小区RSRP差值（即C2I）的均值C2I\_mean、标准差std计算

根据导入的MRO测量报告数据表tbMROData表中主小区和邻小区的参考信号接收功率RSRP，计算主邻小区RSRP差值，该联系是多对多联系。

计算每对主邻小区<SCell, NCell>的RSRP差值C2I，并计算出C2I的均值和标准差。

在求主邻小区对的差值的均值和平均差时，需要对数据分组，即按照Scell和Ncell分组，在MRO表中，是ServingSector和InterferingSector，通过内置函数avg和stdev实现均值和标准差的计算

语句如下：

create procedure Calc\_mean\_std\_pro

as

begin

select ServingSector, InterferingSector, avg(LteScRSRP - LteNcRSRP) as mean, stdev(LteScRSRP - LteNcRSRP) as std into tbMROMean\_Std from tbMROData group by ServingSector, InterferingSector

end

将计算结果存放到名为tbMROMean\_Std的表中，方便计算正太分布概率用

2．主邻小区RSRP差值小于9的概率PrbC2I9计算

根据每对主邻小区RSRP差值的均值和标准差，可以得到正态分布，根据正态分布求出主邻小区RSRP差值小于9的概率Prb9=Pr{C2I<9db}。

3．主邻小区RSRP差值绝对值小于6的概率PrbABS6的计算

根据每对主邻小区RSRP差值的均值和标准差，可以得到正态分布，根据正态分布求出主邻小区RSRP差值绝对值小于6的概率（PrbABS6）。

4．生成新表tbC2Inew

新表包括六个属性：主小区ID、邻小区ID、主邻小区RSRP差值的均值、主邻小区RSRP差值的标准差、主邻小区RSRP差值小于9的概率、主邻小区RSRP差值绝对值小于6的概率。

注意：生成的新表里的数据与tbC2I中的数据值不一样。

我们将2，3，4调整顺序，按照4，2and3的顺序完成:

首先创建tbC2Inew表，语句如下：

create table tbC2INew (

ServingSector nvarchar(50),

InterferingSector nvarchar(50),

mean float,

std float,

Prb9 float,

Prb6 float

)

考虑到需要求正太分布概率，我们需要导入正态分布概率表tbNormStatistics.xlsx表，此表结构为（id, prb），id为0-499，表示正态分布概率表中0.00-4.99的范围，prb为对应0.00-4.99的概率。

对RSRP差值计算正态分布，是一个计算量较大的过程，我们再次创建一个存储过程，用于减少编译时间，语句如下：

create procedure RSRP\_Prb\_pro

as

begin

truncate table tbC2INew --冲表

insert into tbC2INew select ServingSector, InterferingSector, mean, std, 0, 0 from tbMROMean\_Std --默认概率为0

update tbC2INew set Prb9 = (

case

when tbC2INew.Prb9 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (9 - tbC2INew.mean)/tbC2INew.std < 5 and (9 - tbC2INew.mean)/tbC2INew.std >= 0 then (select prb from tbNormStatistics where tbNormStatistics.id = round((9 - tbC2INew.mean)/tbC2INew.std, 2, 0) \* 100)-- 0~4.99，查表

when tbC2INew.Prb9 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (9 - tbC2INew.mean)/tbC2INew.std < 0 then 1 - (select prb from tbNormStatistics where tbNormStatistics.id = abs(round((9 - tbC2INew.mean)/tbC2INew.std, 2, 0)) \* 100)-- <0，1 - 绝对值，查表

when tbC2INew.Prb9 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (9 - tbC2INew.mean)/tbC2INew.std >= 5 then 1 -->5， 1

end

), Prb6 = (

case

when tbC2INew.Prb6 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (6 - tbC2INew.mean)/tbC2INew.std < 5 and (6 - tbC2INew.mean)/tbC2INew.std >= 0 then (select prb from tbNormStatistics where tbNormStatistics.id = round((6 - tbC2INew.mean)/tbC2INew.std, 2, 0) \* 100)-- 0~4.99，查表

when tbC2INew.Prb6 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (6 - tbC2INew.mean)/tbC2INew.std < 0 then 1 - (select prb from tbNormStatistics where tbNormStatistics.id = abs(round((6 - tbC2INew.mean)/tbC2INew.std, 2, 0)) \* 100)-- <0，1 - 绝对值，查表

when tbC2INew.Prb6 = 0 and tbC2INew.std is not null and tbC2INew.std != 0 and (6 - tbC2INew.mean)/tbC2INew.std >= 5 then 1 -->5， 1

end

)

select \* from tbC2INew where Prb9 is not null and Prb6 is not null --由于std有null和0，这些行不能算出正态分布

end

对上述存储过程，在此的具体解释如下：

truncate--为保证数据更新，每次调用时都将原有数据删除

首先对所有的ServingSector和InterferingSector，默认的Prb9和Prb6为0。

然后对这些插入行进行更新Prb9和Prb6：

可以看出RSRP差值的正态分布，不是标准正态分布，要想求这些正太分布概率，就需要转换为求标准正态分布概率(X的正太分布概率 = （X – 均值）/标准差的标准正态分布概率)，再查标准正态分布概率表，具体细节分为：

求Prb9，先求(9-mean) / std，然后比较是否大于4.99，大于4.99的情况，认为其概率为1，否则需要再细分情况考虑，0~4.99范围，则直接查表，查tbNormStatistics表时，0~4.99 \* 100即为id值；< 0的范围，需要用其相反数查表，并1-这个查到的相反数的概率，即为该<0数的概率。

求Prb6的过程同Prb9。

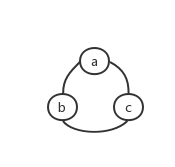
需要注意的是，部分std值为0或者null，这些不能求出正态分布，因此tbC2Inew表中有相当一部分Prb9和Prb6为0。

在调用过程时，使用如下语句：

exec dbo.RSRP\_Prb\_pro

## 3.4 查询重叠覆盖干扰三元组

根据表tbC2Inew，找出所有的小区三元组<a,b,c>（其中a,b,c互为邻小区），生成新表tbC2I3，其中有三个属性，分别是三个小区的小区ID。



使得：

1. PrbABS6(a,b)或PrbABS6(b,a)>=70%;
2. PrbABS6(a,c)或PrbABS6(c,a)>=70%;
3. PrbABS6(b,c)或PrbABS6(c,b)>=70%;

注意：PrbABS6即主邻小区RSRP差值的绝对值小于6的概率。

在根据上述方法计算出的三元组<a, b, c>中，小区a、b、c相互间信号差别不大，没有主导小区，出现严重重叠覆盖，会严重影响通话质量、下载速率等KPI性能指标。

首先创建tbC2I3表，语句如下：

create table tbC2I3 (

a varchar(50),

b varchar(50),

c varchar (50),

PRIMARY KEY (a,b,c)

)

考虑到在后续步骤中需要多次计算两个邻区之间的Prb6，所以先创建一个函数，用于查找给定两个小区的Prb6，创建该函数的语句如下：

create function Prb6GE70(@x varchar(50), @y varchar(50))

returns float

as

begin

declare @rslt float

set @rslt = 0

select @rslt = Prb6 from tbC2INew where ServingSector = @x and InterferingSector = @y and Prb6 >= 0.7

return @rslt

end

函数参数为x，y，分别是两个邻区，若x,y相邻且Prb6不小于0.7，那么返回这个Prb6，否则返回0。

有了这个函数后，我们在查询三元组就可以方便地调用，语句如下：

with sa (ServingSector, InterferingSector) as (

select ServingSector, InterferingSector from tbC2INew

), --a的小区

sb (ServingSector, InterferingSector) as (

select ServingSector, InterferingSector from tbC2INew

), --b的小区

sc (ServingSector, InterferingSector) as (

select ServingSector, InterferingSector from tbC2INew

) --c的小区

insert into tbC2I3 select sa.ServingSector as a, sb.ServingSector as b, sc.ServingSector as c from sa, sb, sc where sa.ServingSector != sb.ServingSector and sb.ServingSector!=sc.ServingSector and sc.ServingSector != sa.ServingSector --不能两两成环

and sa.InterferingSector=sb.ServingSector and sb.InterferingSector=sc.ServingSector and sc.InterferingSector=sa.ServingSector --三个成环

and ((select dbo.Prb6GE70(sa.ServingSector, sb.ServingSector)) != 0 or (select dbo.Prb6GE70(sb.ServingSector, sa.ServingSector)) != 0 ) --概率条件(1)

and ((select dbo.Prb6GE70(sa.ServingSector, sc.ServingSector)) != 0 or (select dbo.Prb6GE70(sc.ServingSector, sa.ServingSector)) != 0 ) --概率条件(2)

and ((select dbo.Prb6GE70(sc.ServingSector, sb.ServingSector)) != 0 or (select dbo.Prb6GE70(sb.ServingSector, sc.ServingSector)) != 0 ) --概率条件(3)

select \* from tbC2I3

sa,sb,sc是对tbC2Inew表的三次查询，仅获得ServingSector和InterferingSector。然后找到构成环形的三个邻区，并且满足Prb6的三个条件，将满足的a,b,c插入到创建好的tbC2I3表中