

Oracle BI по-русски

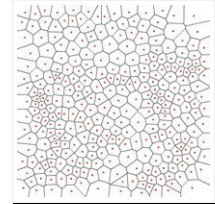


6 дек. 2012 г.

Spatial: диаграмма Вороного (Java)

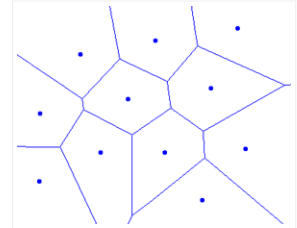
Всем привет!
Сегодня хочу описать решение по генерации диаграммы Вороного с помощью java.

Диаграмма Вороного конечного множества точек S на плоскости представляет такое разбиение плоскости, при котором каждая область этого разбиения образует множество точек, более близких к одному из элементов множества S, чем к любому другому элементу множества.



У диаграммы очень много областей применения, особенно при решении spatial-задач.
/*Например, я с помощью диаграммы Вороного решал задачу кластеризации на карте Санкт-Петербурга */
К сожалению, в Oracle Spatial нет эффективного механизма (я ошибаюсь?) получения данной диаграммы. Да и к тому же Spatial - платная опция...
Поэтому, возможно, приведенное далее решение будет вам интересно!

В решение используется **алгоритм** от Steven Fortune - "метод заметающей прямой".



Само решение большей частью взято отсюда - <http://stackoverflow.com/questions/2346148/fastest-way-to-get-the-set-of-convex-polygons-formed-by-voronoi-line-segments>

И доработано таким образом, что при входных данных:
1) набор гео-точек (например, дома города);
2) ограничивающий точки полигон (например, административная граница города)

- возвращается набор полигонов, представляющих собой ячейки диаграммы Вороного.
Причем ячейки Вороного "обрезаны" ограничивающим полигоном.

В качестве примера рассмотрим формирования ячеек Вороного для Василеостровского района Санкт-Петербурга.

- 1) Скачаем свежие **shape-файлы** по Санкт-Петербургу
- 2) Скачиваем - если вы еще не обзавелись - **Oracle Mapbuilder**
- 3) Запускаем Mapbuilder
(указываем явно кодировку, чтобы не "потерять" кириллические символы)

```
1 | java -Dfile.encoding=UTF8 -XX:MaxPermSize=512m -Xmx1024m -jar mapbuilder.jar
```

- 4) Создаем новую схему в БД для spatial-тестов

```
1 | CREATE USER GEO IDENTIFIED BY GEO DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;  
2 |  
3 | GRANT CONNECT TO GEO;  
4 | GRANT RESOURCE TO GEO;
```

5) Импортируем 2 shape-файла в БД (аналогично **этой**) - boundary-polygon.shp и poi-point.shp в таблицы D_ZONE_POLY_8307 и D_POI_POINT_8307 соответственно.

6) Меняю у загруженных гео-данных в сферическую систему координат (возможно, вы в дальнейшем будете выводить эти данные в Mapviewer/OBIEE11g)

```
1 | create table D_ZONE_POLY_3785 as select * from D_ZONE_POLY_8307;  
2 | update D_ZONE_POLY_3785 set geometry = sdo_cs.transform(geometry, 'USE_SPHERICAL', 3785);  
3 | commit;  
4 |  
5 | delete from user_sdo_geom_metadata where table_name = 'D_ZONE_POLY_3785';  
6 | insert into user_sdo_geom_metadata  
7 | values('D_ZONE_POLY_3785', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_di  
8 | create index D_ZONE_POLY_3785_sidx on D_ZONE_POLY_3785(GEOMETRY) indextype is mdsys.spatial_index;
```

```
1 | create table D_POI_POINT_3785 as select * from D_POI_POINT_8307;  
2 | update D_POI_POINT_3785 set geometry = sdo_cs.transform(geometry, 'USE_SPHERICAL', 3785);  
3 | commit;  
4 |  
5 | delete from user_sdo_geom_metadata where table_name = 'D_POI_POINT_3785';  
6 | insert into user_sdo_geom_metadata  
7 | values('D_POI_POINT_3785', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_di  
8 | create index D_POI_POINT_3785_sidx on D_POI_POINT_3785(GEOMETRY) indextype is mdsys.spatial_index;
```

Translate

Выбрать язык ▾ Технологии GoogleПереводчик

Обо мне

Sergey Sheremeta

Подписаться

64

Просмотреть профиль

Cbox

sofigent1: В Oracle у меня есть хранимая процедура которая в зависимости от параметров ведет запись в временную таблицу которую я потом использую как справочник. Теперь захотел повторить отчет в BI. Он выглядит при грубом приближении как select переименованный со справочником полученным вызовом хранимой процедуры. Думал в BI сделать Аналог с фильтром по другому Аналізу, но не смог передать во внутренний анализ параметр. Как и не смог вовать хранимую процедуру чтоб сделать запись в временную таблицу прямо в БД. Пытался передать параметром сессии значения внутри sub-Анализа, тоже не вышло! Если бы вы могли мне помочь я был бы очень благодарен!!! (име почему-то стало запрещено помянал)

10 Oct 14, 07:10 PM

Jack Carver: Привет! Посмотрите обсуждение тут - http://www.sql.ru/forum/989398/pipelined-function-oracle-business-intelligence-obiee

14 Oct 14, 08:17 PM

Jack Carver: Не советую вам разбираться с временными таблицами. Замучаетесь в отладке. Подозреваю, что обмен данных во временной таблице будет невелик - поэтому на вашем месте использовал бы pipelined-функцию.

14 Oct 14, 08:18 PM

Jack Carver: Эту функцию используйте как физическую таблицу в репозитории. Укажите тип - SELECT. А внутри нечто вроде:

14 Oct 14, 08:22 PM

Jack Carver: (SELECT olo1_1. olo1_2... FROM TABLE(MY_OLOLO_SCHEMA.MY_OLOLO_PKG.my_olo1_pipelined_func(VALUEOF(NO_SESSION.MY_OLOLO_PARAMETER))))

14 Oct 14, 08:22 PM

Jack Carver: (не забудьте запрос обернуть скобками)

14 Oct 14, 08:23 PM

14 пользователей онлайн

обновить

Имя

Своих истек

е-mail / url

OK

помощь · смайлы · профиль

Облако тегов

BI Publisher BIEE 11g BIEE Excel map pivot XSL dashboard parameter XLS XML crosstable mapviewer presentation variable prompt writeback Direct database request Selenium URL bursting cyrillic log multiselect openstreetmap osm perfrom template time-series xslt карты 29 february 40 sec AGO Actions Framework CatalogTreeModel Copy-paste Ctrl-C DOE ELK FUNCTION Firefox GO URL IE Implicit Fact Column Internet Explorer JUnit Jenkins Kibana MSIE PL/SQL PY R RTF Regression Testing SAXParserFactory Spatial Usage Tracking VBA Voronoi WMS Workbook_Open XDO_ XDO_GROUP XDO_SHEET XDO_SHEET_NAME answer balance cache cascading prompts case sensitive chart connection script dashboard list dashboard pages data mining data source dense_rank elasticsearch email filter forecast image javascript jboss lock logout logstash macros monitoring number-rows-spanned obiee only planning prompts proxy rank redline relative path sheet signout socket timeout sub template thumbnail timeout uppercase useless web catalog wsdl xdo:debug xmlhttp:sv2 zabbix zip

Популярные сообщения

- BIP: Excel-шаблоны разметки
- BIEE 11g: Mapviewer + OpenStreetMap
- BIEE: множество страниц дашборда
- BIP: XSLT шаблоны
- BIEE: множество страниц дашборда - часть2
- BI Publisher 11g: отправка zip-результатов по email
- BIEE: сохранение контекста при переходе между информационными панелями
- BIEE: вызов информационных панелей с инициализацией переменных презентаций
- BIP: Число прописью в RTF-шаблоне
- BIEE11g: BI Publisher представления для BI Answer

Архив блога

- 2016 (1)
- 2015 (5)
- 2014 (4)
- 2013 (11)
- ▼ 2012 (7)
 - ▾ декабря (4)
 - BIEE11g: картинки-миниатюры для информационных пан...
 - BIEE 11g: список информационных панелей (PL/SQL)
 - Spatial: диаграмма Вороного (Java)
 - BIEE 11g: каскадно-зависимые приглашения инфопанел...
 - ноября (2)
 - августа (1)
- 2011 (7)
- 2010 (10)
- 2009 (17)

RU блоги

...

7) Вычленим из загруженных данных часть для проведения теста.

Тестовая выборка будет содержать полигон административной границы Василеостровского района Санкт-Петербурга и его дома.

```
1 create table D_ZONE_POLY_TEST as
2 select * from D_ZONE_POLY_3785 t
3 where name = 'Василеостровский район';
4
5 delete from user_sdo_geom_metadata where table_name = 'D_ZONE_POLY_TEST';
6 insert into user_sdo_geom_metadata
7 values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_
8
9 create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index;
```

В таблицу D_POI_POINT_TEST сохраняем лишь дома, принадлежащие выбранному району города.

```
1 create table D_POI_POINT_TEST as
2 select p.*
3 from D_POI_POINT_3785 p, D_ZONE_POLY_TEST z
4 where SDO_CONTAINS(z.geometry, p.geometry) = 'TRUE';
5
6 delete from user_sdo_geom_metadata where table_name = 'D_POI_POINT_TEST';
7 insert into user_sdo_geom_metadata
8 values('D_POI_POINT_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_c
9
10 create index D_POI_POINT_TEST_sidx on D_POI_POINT_TEST(GEOMETRY) indextype is mdsys.spatial_index;
```

8) Создадим таблицу для хранения результатов теста

```
1 create table D_VORONOI_POLY(GEOMETRY MDSYS.SDO_GEOMETRY, OSM_ID NUMBER, NEIGHBORS VARCHAR2(4000));
2
3 delete from user_sdo_geom_metadata where table_name = 'D_VORONOI_POLY';
4 insert into user_sdo_geom_metadata
5 values('D_VORONOI_POLY', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_
6
7 create index D_VORONOI_POLY_sidx on D_VORONOI_POLY(GEOMETRY) indextype is mdsys.spatial_index;
```

9) Скачаем архив с java-проектом генерации диаграммы Вороного.

В главном файле проекта - src\ru\servplus\geom\Voronoi.java - следует изменить настройки JDBC-доступа к тестовой схеме БД.

```
1 package ru.servplus.geom;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.util.Iterator;
8 import java.util.LinkedList;
9 import java.util.ListIterator;
10 import java.util.PriorityQueue;
11 import java.util.TreeSet;
12
13 import oracle.jdbc.*;
14 import oracle.spatial.geometry.JGeometry;
15
16 import com.vividsolutions.jts.geom.Coordinate;
17 import com.vividsolutions.jts.geom.Geometry;
18 import com.vividsolutions.jts.geom.GeometryFactory;
19 import com.vividsolutions.jts.geom.LineString;
20
21 import com.vividsolutions.jts.geom.LinearRing;
22
23
24 public class Voronoi {
25     boolean debug = false;
26
27     Geometry zonePoly;
28     GeometryFactory geometryFactory;
29
30     double xmin;
31     double ymin;
32     double xmax;
33     double ymax;
34     double width;
35     double height;
36
37     Point topleft;
38     Point topright;
39     Point bottomleft;
40     Point bottomright;
41
42     Connection conn = null;
43
44     // The set of points that control the centers of the cells
45     private LinkedList<Point> pts;
46     // A list of line segments that defines where the cells are divided
47     private LinkedList<Edge> output;
48     // Special list of points that are redundant and need to be removed in post processing
49     private LinkedList<Point> mechs;
50     // The sites that have not yet been processed, in ascending order of X coordinate
51     private PriorityQueue<Point> sites;
52     // Possible upcoming circle events in ascending order of X coordinate
53     private PriorityQueue<CircleEvent> events;
54     // The root of the binary search tree of the parabolic wave front
55     private Arc root;
56
57
58     public static void main(String[] args) {
59         System.out.println("start!");
60         try {
61             Voronoi vor = new Voronoi();
62             vor.execute();
63         }
64         catch (Exception e)
65         {
66             e.printStackTrace();
67         }
68
69         System.out.println("end!");
70     }
71
72     public Voronoi() throws Exception{
73
74         //Init the lists
75         pts = new LinkedList<Point>();
76         output = new LinkedList<Edge>();
77         mechs = new LinkedList<Point>();
78         sites = new PriorityQueue<Point>(100);
79         events = new PriorityQueue<CircleEvent>(100);
80
81         JGeometry geom = null;
82         PreparedStatement pstmt = null;
83         ResultSet rs = null;
84         oracle.sql.STRUCT st = null;
85         Point p;
86         int key;
87
88         String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:ORCL";
89         String jdbcUser = "GEO";
90         String jdbcPwd = "GEO";
91
92         Class.forName("oracle.jdbc.OracleDriver").newInstance();
93         this.conn = DriverManager.getConnection(jdbcUrl, jdbcUser, jdbcPwd);
94
95         if(conn == null)
96         {
97             System.out.print("cannot get connection!");
98         }
99
100         //calculate zone MBR
101         pstmt = conn.prepareStatement(
102             "select max(decode(t.id, 1, t.x, null)) x_min, "+
103             "      max(decode(t.id, 2, t.x, null)) x_max, "+
104             "      max(decode(t.id, 1, t.y, null)) y_min, "+
105             "      max(decode(t.id, 2, t.y, null)) y_max "+
106             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
107
108         rs = pstmt.executeQuery();
109         while(rs.next())
110         {
111             xmin = rs.getDouble(1);
112             xmax = rs.getDouble(2);
113             ymin = rs.getDouble(3);
114             ymax = rs.getDouble(4);
115             width = xmax - xmin;
116             height = ymax - ymin;
117
118             topleft = new Point(xmin, ymin);
119             topright = new Point(xmax, ymin);
120             bottomleft = new Point(xmin, ymax);
121             bottomright = new Point(xmax, ymax);
122
123             pts.add(topleft);
124             pts.add(topright);
125             pts.add(bottomleft);
126             pts.add(bottomright);
127
128             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
129             zonePoly = geometryFactory.createGeometry(geom);
130
131             //Save the zone MBR to the database
132             pstmt = conn.prepareStatement(
133                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
134             pstmt.executeUpdate();
135
136             pstmt = conn.prepareStatement(
137                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
138             pstmt.executeUpdate();
139         }
140     }
141
142     void execute() throws Exception{
143         //Get the zone MBR from the database
144         pstmt = conn.prepareStatement(
145             "select max(decode(t.id, 1, t.x, null)) x_min, "+
146             "      max(decode(t.id, 2, t.x, null)) x_max, "+
147             "      max(decode(t.id, 1, t.y, null)) y_min, "+
148             "      max(decode(t.id, 2, t.y, null)) y_max "+
149             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
150         rs = pstmt.executeQuery();
151         while(rs.next())
152         {
153             xmin = rs.getDouble(1);
154             xmax = rs.getDouble(2);
155             ymin = rs.getDouble(3);
156             ymax = rs.getDouble(4);
157             width = xmax - xmin;
158             height = ymax - ymin;
159
160             topleft = new Point(xmin, ymin);
161             topright = new Point(xmax, ymin);
162             bottomleft = new Point(xmin, ymax);
163             bottomright = new Point(xmax, ymax);
164
165             pts.add(topleft);
166             pts.add(topright);
167             pts.add(bottomleft);
168             pts.add(bottomright);
169
170             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
171             zonePoly = geometryFactory.createGeometry(geom);
172
173             //Save the zone MBR to the database
174             pstmt = conn.prepareStatement(
175                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
176             pstmt.executeUpdate();
177
178             pstmt = conn.prepareStatement(
179                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
180             pstmt.executeUpdate();
181         }
182     }
183
184     void execute2() throws Exception{
185         //Get the zone MBR from the database
186         pstmt = conn.prepareStatement(
187             "select max(decode(t.id, 1, t.x, null)) x_min, "+
188             "      max(decode(t.id, 2, t.x, null)) x_max, "+
189             "      max(decode(t.id, 1, t.y, null)) y_min, "+
190             "      max(decode(t.id, 2, t.y, null)) y_max "+
191             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
192         rs = pstmt.executeQuery();
193         while(rs.next())
194         {
195             xmin = rs.getDouble(1);
196             xmax = rs.getDouble(2);
197             ymin = rs.getDouble(3);
198             ymax = rs.getDouble(4);
199             width = xmax - xmin;
200             height = ymax - ymin;
201
202             topleft = new Point(xmin, ymin);
203             topright = new Point(xmax, ymin);
204             bottomleft = new Point(xmin, ymax);
205             bottomright = new Point(xmax, ymax);
206
207             pts.add(topleft);
208             pts.add(topright);
209             pts.add(bottomleft);
210             pts.add(bottomright);
211
212             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
213             zonePoly = geometryFactory.createGeometry(geom);
214
215             //Save the zone MBR to the database
216             pstmt = conn.prepareStatement(
217                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
218             pstmt.executeUpdate();
219
220             pstmt = conn.prepareStatement(
221                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
222             pstmt.executeUpdate();
223         }
224     }
225
226     void execute3() throws Exception{
227         //Get the zone MBR from the database
228         pstmt = conn.prepareStatement(
229             "select max(decode(t.id, 1, t.x, null)) x_min, "+
230             "      max(decode(t.id, 2, t.x, null)) x_max, "+
231             "      max(decode(t.id, 1, t.y, null)) y_min, "+
232             "      max(decode(t.id, 2, t.y, null)) y_max "+
233             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
234         rs = pstmt.executeQuery();
235         while(rs.next())
236         {
237             xmin = rs.getDouble(1);
238             xmax = rs.getDouble(2);
239             ymin = rs.getDouble(3);
240             ymax = rs.getDouble(4);
241             width = xmax - xmin;
242             height = ymax - ymin;
243
244             topleft = new Point(xmin, ymin);
245             topright = new Point(xmax, ymin);
246             bottomleft = new Point(xmin, ymax);
247             bottomright = new Point(xmax, ymax);
248
249             pts.add(topleft);
250             pts.add(topright);
251             pts.add(bottomleft);
252             pts.add(bottomright);
253
254             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
255             zonePoly = geometryFactory.createGeometry(geom);
256
257             //Save the zone MBR to the database
258             pstmt = conn.prepareStatement(
259                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
260             pstmt.executeUpdate();
261
262             pstmt = conn.prepareStatement(
263                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
264             pstmt.executeUpdate();
265         }
266     }
267
268     void execute4() throws Exception{
269         //Get the zone MBR from the database
270         pstmt = conn.prepareStatement(
271             "select max(decode(t.id, 1, t.x, null)) x_min, "+
272             "      max(decode(t.id, 2, t.x, null)) x_max, "+
273             "      max(decode(t.id, 1, t.y, null)) y_min, "+
274             "      max(decode(t.id, 2, t.y, null)) y_max "+
275             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
276         rs = pstmt.executeQuery();
277         while(rs.next())
278         {
279             xmin = rs.getDouble(1);
280             xmax = rs.getDouble(2);
281             ymin = rs.getDouble(3);
282             ymax = rs.getDouble(4);
283             width = xmax - xmin;
284             height = ymax - ymin;
285
286             topleft = new Point(xmin, ymin);
287             topright = new Point(xmax, ymin);
288             bottomleft = new Point(xmin, ymax);
289             bottomright = new Point(xmax, ymax);
290
291             pts.add(topleft);
292             pts.add(topright);
293             pts.add(bottomleft);
294             pts.add(bottomright);
295
296             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
297             zonePoly = geometryFactory.createGeometry(geom);
298
299             //Save the zone MBR to the database
300             pstmt = conn.prepareStatement(
301                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
302             pstmt.executeUpdate();
303
304             pstmt = conn.prepareStatement(
305                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
306             pstmt.executeUpdate();
307         }
308     }
309
310     void execute5() throws Exception{
311         //Get the zone MBR from the database
312         pstmt = conn.prepareStatement(
313             "select max(decode(t.id, 1, t.x, null)) x_min, "+
314             "      max(decode(t.id, 2, t.x, null)) x_max, "+
315             "      max(decode(t.id, 1, t.y, null)) y_min, "+
316             "      max(decode(t.id, 2, t.y, null)) y_max "+
317             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
318         rs = pstmt.executeQuery();
319         while(rs.next())
320         {
321             xmin = rs.getDouble(1);
322             xmax = rs.getDouble(2);
323             ymin = rs.getDouble(3);
324             ymax = rs.getDouble(4);
325             width = xmax - xmin;
326             height = ymax - ymin;
327
328             topleft = new Point(xmin, ymin);
329             topright = new Point(xmax, ymin);
330             bottomleft = new Point(xmin, ymax);
331             bottomright = new Point(xmax, ymax);
332
333             pts.add(topleft);
334             pts.add(topright);
335             pts.add(bottomleft);
336             pts.add(bottomright);
337
338             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
339             zonePoly = geometryFactory.createGeometry(geom);
340
341             //Save the zone MBR to the database
342             pstmt = conn.prepareStatement(
343                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
344             pstmt.executeUpdate();
345
346             pstmt = conn.prepareStatement(
347                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
348             pstmt.executeUpdate();
349         }
350     }
351
352     void execute6() throws Exception{
353         //Get the zone MBR from the database
354         pstmt = conn.prepareStatement(
355             "select max(decode(t.id, 1, t.x, null)) x_min, "+
356             "      max(decode(t.id, 2, t.x, null)) x_max, "+
357             "      max(decode(t.id, 1, t.y, null)) y_min, "+
358             "      max(decode(t.id, 2, t.y, null)) y_max "+
359             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
360         rs = pstmt.executeQuery();
361         while(rs.next())
362         {
363             xmin = rs.getDouble(1);
364             xmax = rs.getDouble(2);
365             ymin = rs.getDouble(3);
366             ymax = rs.getDouble(4);
367             width = xmax - xmin;
368             height = ymax - ymin;
369
370             topleft = new Point(xmin, ymin);
371             topright = new Point(xmax, ymin);
372             bottomleft = new Point(xmin, ymax);
373             bottomright = new Point(xmax, ymax);
374
375             pts.add(topleft);
376             pts.add(topright);
377             pts.add(bottomleft);
378             pts.add(bottomright);
379
380             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
381             zonePoly = geometryFactory.createGeometry(geom);
382
383             //Save the zone MBR to the database
384             pstmt = conn.prepareStatement(
385                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
386             pstmt.executeUpdate();
387
388             pstmt = conn.prepareStatement(
389                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
390             pstmt.executeUpdate();
391         }
392     }
393
394     void execute7() throws Exception{
395         //Get the zone MBR from the database
396         pstmt = conn.prepareStatement(
397             "select max(decode(t.id, 1, t.x, null)) x_min, "+
398             "      max(decode(t.id, 2, t.x, null)) x_max, "+
399             "      max(decode(t.id, 1, t.y, null)) y_min, "+
400             "      max(decode(t.id, 2, t.y, null)) y_max "+
401             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
402         rs = pstmt.executeQuery();
403         while(rs.next())
404         {
405             xmin = rs.getDouble(1);
406             xmax = rs.getDouble(2);
407             ymin = rs.getDouble(3);
408             ymax = rs.getDouble(4);
409             width = xmax - xmin;
410             height = ymax - ymin;
411
412             topleft = new Point(xmin, ymin);
413             topright = new Point(xmax, ymin);
414             bottomleft = new Point(xmin, ymax);
415             bottomright = new Point(xmax, ymax);
416
417             pts.add(topleft);
418             pts.add(topright);
419             pts.add(bottomleft);
420             pts.add(bottomright);
421
422             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
423             zonePoly = geometryFactory.createGeometry(geom);
424
425             //Save the zone MBR to the database
426             pstmt = conn.prepareStatement(
427                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
428             pstmt.executeUpdate();
429
430             pstmt = conn.prepareStatement(
431                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
432             pstmt.executeUpdate();
433         }
434     }
435
436     void execute8() throws Exception{
437         //Get the zone MBR from the database
438         pstmt = conn.prepareStatement(
439             "select max(decode(t.id, 1, t.x, null)) x_min, "+
440             "      max(decode(t.id, 2, t.x, null)) x_max, "+
441             "      max(decode(t.id, 1, t.y, null)) y_min, "+
442             "      max(decode(t.id, 2, t.y, null)) y_max "+
443             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
444         rs = pstmt.executeQuery();
445         while(rs.next())
446         {
447             xmin = rs.getDouble(1);
448             xmax = rs.getDouble(2);
449             ymin = rs.getDouble(3);
450             ymax = rs.getDouble(4);
451             width = xmax - xmin;
452             height = ymax - ymin;
453
454             topleft = new Point(xmin, ymin);
455             topright = new Point(xmax, ymin);
456             bottomleft = new Point(xmin, ymax);
457             bottomright = new Point(xmax, ymax);
458
459             pts.add(topleft);
460             pts.add(topright);
461             pts.add(bottomleft);
462             pts.add(bottomright);
463
464             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
465             zonePoly = geometryFactory.createGeometry(geom);
466
467             //Save the zone MBR to the database
468             pstmt = conn.prepareStatement(
469                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
470             pstmt.executeUpdate();
471
472             pstmt = conn.prepareStatement(
473                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
474             pstmt.executeUpdate();
475         }
476     }
477
478     void execute9() throws Exception{
479         //Get the zone MBR from the database
480         pstmt = conn.prepareStatement(
481             "select max(decode(t.id, 1, t.x, null)) x_min, "+
482             "      max(decode(t.id, 2, t.x, null)) x_max, "+
483             "      max(decode(t.id, 1, t.y, null)) y_min, "+
484             "      max(decode(t.id, 2, t.y, null)) y_max "+
485             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
486         rs = pstmt.executeQuery();
487         while(rs.next())
488         {
489             xmin = rs.getDouble(1);
490             xmax = rs.getDouble(2);
491             ymin = rs.getDouble(3);
492             ymax = rs.getDouble(4);
493             width = xmax - xmin;
494             height = ymax - ymin;
495
496             topleft = new Point(xmin, ymin);
497             topright = new Point(xmax, ymin);
498             bottomleft = new Point(xmin, ymax);
499             bottomright = new Point(xmax, ymax);
500
501             pts.add(topleft);
502             pts.add(topright);
503             pts.add(bottomleft);
504             pts.add(bottomright);
505
506             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
507             zonePoly = geometryFactory.createGeometry(geom);
508
509             //Save the zone MBR to the database
510             pstmt = conn.prepareStatement(
511                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
512             pstmt.executeUpdate();
513
514             pstmt = conn.prepareStatement(
515                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
516             pstmt.executeUpdate();
517         }
518     }
519
520     void execute10() throws Exception{
521         //Get the zone MBR from the database
522         pstmt = conn.prepareStatement(
523             "select max(decode(t.id, 1, t.x, null)) x_min, "+
524             "      max(decode(t.id, 2, t.x, null)) x_max, "+
525             "      max(decode(t.id, 1, t.y, null)) y_min, "+
526             "      max(decode(t.id, 2, t.y, null)) y_max "+
527             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
528         rs = pstmt.executeQuery();
529         while(rs.next())
530         {
531             xmin = rs.getDouble(1);
532             xmax = rs.getDouble(2);
533             ymin = rs.getDouble(3);
534             ymax = rs.getDouble(4);
535             width = xmax - xmin;
536             height = ymax - ymin;
537
538             topleft = new Point(xmin, ymin);
539             topright = new Point(xmax, ymin);
540             bottomleft = new Point(xmin, ymax);
541             bottomright = new Point(xmax, ymax);
542
543             pts.add(topleft);
544             pts.add(topright);
545             pts.add(bottomleft);
546             pts.add(bottomright);
547
548             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
549             zonePoly = geometryFactory.createGeometry(geom);
550
551             //Save the zone MBR to the database
552             pstmt = conn.prepareStatement(
553                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
554             pstmt.executeUpdate();
555
556             pstmt = conn.prepareStatement(
557                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
558             pstmt.executeUpdate();
559         }
560     }
561
562     void execute11() throws Exception{
563         //Get the zone MBR from the database
564         pstmt = conn.prepareStatement(
565             "select max(decode(t.id, 1, t.x, null)) x_min, "+
566             "      max(decode(t.id, 2, t.x, null)) x_max, "+
567             "      max(decode(t.id, 1, t.y, null)) y_min, "+
568             "      max(decode(t.id, 2, t.y, null)) y_max "+
569             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
570         rs = pstmt.executeQuery();
571         while(rs.next())
572         {
573             xmin = rs.getDouble(1);
574             xmax = rs.getDouble(2);
575             ymin = rs.getDouble(3);
576             ymax = rs.getDouble(4);
577             width = xmax - xmin;
578             height = ymax - ymin;
579
580             topleft = new Point(xmin, ymin);
581             topright = new Point(xmax, ymin);
582             bottomleft = new Point(xmin, ymax);
583             bottomright = new Point(xmax, ymax);
584
585             pts.add(topleft);
586             pts.add(topright);
587             pts.add(bottomleft);
588             pts.add(bottomright);
589
590             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
591             zonePoly = geometryFactory.createGeometry(geom);
592
593             //Save the zone MBR to the database
594             pstmt = conn.prepareStatement(
595                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
596             pstmt.executeUpdate();
597
598             pstmt = conn.prepareStatement(
599                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
600             pstmt.executeUpdate();
601         }
602     }
603
604     void execute12() throws Exception{
605         //Get the zone MBR from the database
606         pstmt = conn.prepareStatement(
607             "select max(decode(t.id, 1, t.x, null)) x_min, "+
608             "      max(decode(t.id, 2, t.x, null)) x_max, "+
609             "      max(decode(t.id, 1, t.y, null)) y_min, "+
610             "      max(decode(t.id, 2, t.y, null)) y_max "+
611             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
612         rs = pstmt.executeQuery();
613         while(rs.next())
614         {
615             xmin = rs.getDouble(1);
616             xmax = rs.getDouble(2);
617             ymin = rs.getDouble(3);
618             ymax = rs.getDouble(4);
619             width = xmax - xmin;
620             height = ymax - ymin;
621
622             topleft = new Point(xmin, ymin);
623             topright = new Point(xmax, ymin);
624             bottomleft = new Point(xmin, ymax);
625             bottomright = new Point(xmax, ymax);
626
627             pts.add(topleft);
628             pts.add(topright);
629             pts.add(bottomleft);
630             pts.add(bottomright);
631
632             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
633             zonePoly = geometryFactory.createGeometry(geom);
634
635             //Save the zone MBR to the database
636             pstmt = conn.prepareStatement(
637                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
638             pstmt.executeUpdate();
639
640             pstmt = conn.prepareStatement(
641                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
642             pstmt.executeUpdate();
643         }
644     }
645
646     void execute13() throws Exception{
647         //Get the zone MBR from the database
648         pstmt = conn.prepareStatement(
649             "select max(decode(t.id, 1, t.x, null)) x_min, "+
650             "      max(decode(t.id, 2, t.x, null)) x_max, "+
651             "      max(decode(t.id, 1, t.y, null)) y_min, "+
652             "      max(decode(t.id, 2, t.y, null)) y_max "+
653             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
654         rs = pstmt.executeQuery();
655         while(rs.next())
656         {
657             xmin = rs.getDouble(1);
658             xmax = rs.getDouble(2);
659             ymin = rs.getDouble(3);
660             ymax = rs.getDouble(4);
661             width = xmax - xmin;
662             height = ymax - ymin;
663
664             topleft = new Point(xmin, ymin);
665             topright = new Point(xmax, ymin);
666             bottomleft = new Point(xmin, ymax);
667             bottomright = new Point(xmax, ymax);
668
669             pts.add(topleft);
670             pts.add(topright);
671             pts.add(bottomleft);
672             pts.add(bottomright);
673
674             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
675             zonePoly = geometryFactory.createGeometry(geom);
676
677             //Save the zone MBR to the database
678             pstmt = conn.prepareStatement(
679                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
680             pstmt.executeUpdate();
681
682             pstmt = conn.prepareStatement(
683                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
684             pstmt.executeUpdate();
685         }
686     }
687
688     void execute14() throws Exception{
689         //Get the zone MBR from the database
690         pstmt = conn.prepareStatement(
691             "select max(decode(t.id, 1, t.x, null)) x_min, "+
692             "      max(decode(t.id, 2, t.x, null)) x_max, "+
693             "      max(decode(t.id, 1, t.y, null)) y_min, "+
694             "      max(decode(t.id, 2, t.y, null)) y_max "+
695             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
696         rs = pstmt.executeQuery();
697         while(rs.next())
698         {
699             xmin = rs.getDouble(1);
700             xmax = rs.getDouble(2);
701             ymin = rs.getDouble(3);
702             ymax = rs.getDouble(4);
703             width = xmax - xmin;
704             height = ymax - ymin;
705
706             topleft = new Point(xmin, ymin);
707             topright = new Point(xmax, ymin);
708             bottomleft = new Point(xmin, ymax);
709             bottomright = new Point(xmax, ymax);
710
711             pts.add(topleft);
712             pts.add(topright);
713             pts.add(bottomleft);
714             pts.add(bottomright);
715
716             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
717             zonePoly = geometryFactory.createGeometry(geom);
718
719             //Save the zone MBR to the database
720             pstmt = conn.prepareStatement(
721                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
722             pstmt.executeUpdate();
723
724             pstmt = conn.prepareStatement(
725                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
726             pstmt.executeUpdate();
727         }
728     }
729
730     void execute15() throws Exception{
731         //Get the zone MBR from the database
732         pstmt = conn.prepareStatement(
733             "select max(decode(t.id, 1, t.x, null)) x_min, "+
734             "      max(decode(t.id, 2, t.x, null)) x_max, "+
735             "      max(decode(t.id, 1, t.y, null)) y_min, "+
736             "      max(decode(t.id, 2, t.y, null)) y_max "+
737             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
738         rs = pstmt.executeQuery();
739         while(rs.next())
740         {
741             xmin = rs.getDouble(1);
742             xmax = rs.getDouble(2);
743             ymin = rs.getDouble(3);
744             ymax = rs.getDouble(4);
745             width = xmax - xmin;
746             height = ymax - ymin;
747
748             topleft = new Point(xmin, ymin);
749             topright = new Point(xmax, ymin);
750             bottomleft = new Point(xmin, ymax);
751             bottomright = new Point(xmax, ymax);
752
753             pts.add(topleft);
754             pts.add(topright);
755             pts.add(bottomleft);
756             pts.add(bottomright);
757
758             geom = geometryFactory.createPolygon(pts.toArray(new Point[pts.size()]));
759             zonePoly = geometryFactory.createGeometry(geom);
760
761             //Save the zone MBR to the database
762             pstmt = conn.prepareStatement(
763                 "insert into user_sdo_geom_metadata values('D_ZONE_POLY_TEST', 'GEOMETRY', sdo_dim_array(sdo_dim_element('X', -20037508.3427, 20037508.3427, 0.05), sdo_dim_element('Y', -20037508.3427, 20037508.3427, 0.05)))");
764             pstmt.executeUpdate();
765
766             pstmt = conn.prepareStatement(
767                 "create index D_ZONE_POLY_TEST_sidx on D_ZONE_POLY_TEST(GEOMETRY) indextype is mdsys.spatial_index");
768             pstmt.executeUpdate();
769         }
770     }
771
772     void execute16() throws Exception{
773         //Get the zone MBR from the database
774         pstmt = conn.prepareStatement(
775             "select max(decode(t.id, 1, t.x, null)) x_min, "+
776             "      max(decode(t.id, 2, t.x, null)) x_max, "+
777             "      max(decode(t.id, 1, t.y, null)) y_min, "+
778             "      max(decode(t.id, 2, t.y, null)) y_max "+
779             "      from TABLE (select sdo_util.getvertices(SDO_GEOM.SDO_MBR(geometry)) from D_ZONE_POLY_TEST) t");
780         rs = pstmt.executeQuery();
7
```

```

109 rs = pstmt.executeQuery();
110 while (rs.next()){
111     xmin = rs.getDouble("X_MIN") - 1000;
112     ymin = rs.getDouble("Y_MIN") - 1000;
113     xmax = rs.getDouble("X_MAX") + 1000;
114     ymax = rs.getDouble("Y_MAX") + 1000;
115 }
116 height = ymax - ymin;
117 width = xmax - xmin;
118 pstmt.close();
119
120 topleft = new Point(xmin,ymin);
121 topright = new Point(xmax,ymin);
122 bottomleft = new Point(xmin,ymax);
123 bottomright = new Point(xmax,ymax);
124
125 //get zone Points
126 pstmt = conn.prepareStatement(
127     "select OSM_ID, GEOMETRY " +
128     "from D_POI_POINT_TEST " +
129     "where I=1 ");
130 rs = pstmt.executeQuery();
131
132 while (rs.next())
133 {
134     st = (oracle.sql.STRUCT) rs.getObject("GEOMETRY");
135     key = rs.getInt("OSM_ID");
136     geom = JGeometry.load(st);
137     double[] coords = geom.getPoint();
138     p = new Point(coords[0],coords[1]);
139     p.key = key;
140     pts.add(p);
141 }
142 pstmt.close();
143
144 ListIterator<Point> points = pts.listIterator(0);
145
146 //get zone polygon
147 geometryFactory = new GeometryFactory();
148
149 pstmt = conn.prepareStatement("select GEOMETRY from D_ZONE_POLY_TEST ");
150 rs = pstmt.executeQuery();
151
152 zonePoly = null;
153 LinearRing zoneLr;
154 while (rs.next())
155 {
156     st = (oracle.sql.STRUCT) rs.getObject("GEOMETRY");
157     geom = JGeometry.load(st);
158     Coordinate[] coords = new Coordinate[geom.getNumPoints()];
159     for (int i=0; i<geom.getNumPoints(); i++){
160         coords[i] = new Coordinate(geom.getOrdinatesArray()[i*2],geom.getOrdinatesArray()[i*2 + 1]);
161     }
162     zoneLr = geometryFactory.createLinearRing(coords);
163     zonePoly = geometryFactory.createPolygon(zoneLr, null);
164 }
165 pstmt.close();
166 }
167
168 public void execute() throws Exception{
169
170     JGeometry geom = null;
171     PreparedStatement pstmt = null;
172     //ResultSet rs = null;
173     oracle.sql.STRUCT st = null;
174     Point p;
175
176     if(conn == null)
177     {
178         System.out.print("cannot get connection!");
179     }
180
181     // run fortune's algorithm to get new line segments
182     runFortune(pts);
183
184     APoint v,ov,frst;
185     Point n;
186     StringBuffer neighborsConcat;
187     String neighborsConcatStr = null;
188     ListIterator<Point> siteit = pts.listIterator(0);
189
190     pstmt = conn.prepareStatement("DELETE FROM D_VORONOI_POLY");
191     pstmt.execute();
192
193     pstmt = conn.prepareStatement("INSERT INTO D_VORONOI_POLY(GEOMETRY, OSM_ID, NEIGHBORS) VALUES(?,?,?)");
194     ((OraclePreparedStatement)pstmt).setExecuteBatch(pts.size());
195
196     while (siteit.hasNext()) {
197         p = siteit.next();
198         if (p.sverts != null){
199             Iterator<APoint> vert = p.sverts.iterator();
200             if (vert.hasNext()){
201                 v = vert.next();
202                 double[] ring = new double[(p.sverts.size() + 1) * 2];
203                 int f = 0;
204                 frst = v;
205                 ring[f++] = frst.x;
206                 ring[f++] = frst.y;
207                 while (vert.hasNext()) {
208                     ov = v;
209                     v = vert.next();
210                     ring[f++] = v.x;
211                     ring[f++] = v.y;
212                 }
213                 ring[f++] = frst.x;
214                 ring[f++] = frst.y;
215                 Coordinate[] coords = new Coordinate[ring.length/2];
216                 for (int i=0; i<ring.length/2; i++){
217                     coords[i] = new Coordinate(ring[i*2],ring[i*2 + 1]);
218                 }
219                 LinearRing cellLr = geometryFactory.createLinearRing(coords);
220                 Geometry cellPoly = geometryFactory.createPolygon(cellLr, null);
221                 Geometry cellIntersect = zonePoly.intersection(cellPoly);
222                 if (cellIntersect != null) {
223                     Object[] intersectRings = new Object[cellIntersect.getNumGeometries()];
224                     for(int zz=0; zz<cellIntersect.getNumGeometries(); zz++){
225                         Coordinate[] intersectCoord = cellIntersect.getGeometryN(zz).getCoordinates();
226                         double[] intersectRing = new double[intersectCoord.length * 2];
227                         for (int i=0; i< intersectCoord.length;i++){
228                             intersectRing[i*2] = intersectCoord[i].x;
229                             intersectRing[i*2+1] = intersectCoord[i].y;
230                         }
231                         intersectRings[zz] = intersectRing;
232                     }
233                     try {
234                         geom = JGeometry.createLinearPolygon(intersectRings, 2, 3785);
235                     }
236                     if ( intersectRings.length > 1 && geom.getType() == 3 ) {
237                         geom.setType( JGeometry.GTYPE_MULTIPOLYGON );
238                         //System.out.println("GTYPE_MULTIPOLYGON - "+p.key);
239                     }
240                     st = JGeometry.store(geom, conn);
241                     pstmt.setObject(1, st);
242                     pstmt.setObject(2, p.key);
243                 }
244             }
245         }
246     }

```

```

265     } catch (RuntimeException e) {
266         e.printStackTrace();
267         System.out.println(p.key);
268     }
269
270     neighborsConcat = new StringBuffer();
271
272     Iterator<Point> neighbors = p.neighbors.iterator();
273
274     while (neighbors.hasNext()){
275         n = neighbors.next();
276         neighborsConcat.append(n.key+",");
277     }
278     if (neighborsConcat.length() > 0)
279         neighborsConcatStr = neighborsConcat.substring(0, neighborsConcat.length()-1);
280     pstmt.setObject(3, neighborsConcatStr);
281
282     pstmt.executeUpdate();
283
284     }
285     }
286 }
287
288 ((OraclePreparedStatement)pstmt).sendBatch(); // JDBC sends the queued request
289
290 conn.commit();
291
292 pstmt.close();
293 }
294
295
296 void runFortune(LinkedList<Point> pts) {
297
298     sites.clear();
299     events.clear();
300     output.clear();
301     mechs.clear();
302     root = null;
303
304     ListIterator<Point> i = pts.listIterator(0);
305     Point meh;
306     while (i.hasNext()) {
307         meh = i.next();
308         sites.offer(meh);
309         if (meh.sverts == null)
310             meh.sverts = new TreeSet<APoint>();
311         meh.sverts.clear();
312         if (meh.neighbors == null)
313             meh.neighbors = new TreeSet<Point>();
314         meh.neighbors.clear();
315     }
316
317     // Process the queues; select the top element with smaller x coordinate.
318     while (sites.size() > 0) {
319         if ((events.size() > 0) && ((events.peek().xpos) <= (sites.peek().x))) {
320             processCircleEvent(events.poll());
321         } else {
322             //process a site event by adding a curve to the parabolic front
323             frontInsert(sites.poll());
324         }
325     }
326
327     // After all points are processed, do the remaining circle events.
328     while (events.size() > 0) {
329         processCircleEvent(events.poll());
330     }
331
332     // Clean up dangling edges.
333     finishEdges();
334
335     //make polygons from edges
336     polyFinish();
337 }
338
339 private void processCircleEvent(CircleEvent event) {
340     if (event.valid) {
341         Arc parc = event.a;
342
343         //start a new edge
344         Edge edgy = new Edge(event.p, parc.prev.focus, parc.next.focus);
345
346         // Remove the associated arc from the front;
347         if (parc.prev != null) {
348             parc.prev.next = parc.next;
349             parc.prev.edge1 = edgy;
350         }
351         if (parc.next != null) {
352             parc.next.prev = parc.prev;
353             parc.next.edge0 = edgy;
354         }
355
356         // Finish the edges before and after this arc.
357         if (parc.edge0 != null) {
358             parc.edge0.finish(event.p);
359         }
360         if (parc.edge1 != null) {
361             parc.edge1.finish(event.p);
362         }
363
364         // Recheck circle events on either side of p:
365         if (parc.prev != null)
366             checkCircleEvent(parc.prev, event.xpos);
367         if (parc.next != null)
368             checkCircleEvent(parc.next, event.xpos);
369     }
370 }
371
372 void frontInsert(Point focus) {
373     if (root == null) {
374         root = new Arc(focus);
375         return;
376     }
377
378     Arc parc = root;
379     while (parc != null) {
380         CircleResultPack rez = intersect(focus, parc);
381         if (rez.valid) {
382             // New parabola intersects parc. If necessary, duplicate parc.
383
384             if (parc.next != null) {
385                 CircleResultPack rez2 = intersect(focus, parc.next);
386                 if (rez2.valid) {
387                     Arc bla = new Arc(parc.focus);
388                     bla.prev = parc;
389                     bla.next = parc.next;
390                     parc.next.prev = bla;
391                     parc.next = bla;
392                 }
393             }
394             else {
395                 parc.next = new Arc(parc.focus);
396                 parc.next.prev = parc;
397             }
398             parc.next.edge1 = parc.edge1;
399
400             // Add new arc between parc and parc.next.
401             Arc bla = new Arc(focus);
402             bla.prev = parc;
403             bla.next = parc.next;
404             parc.next.prev = bla;
405             parc.next = bla;
406
407             parc = parc.next; // Now parc points to the new arc.
408
409             // Add new half-edges connected to parc's endpoints.
410             parc.edge0 = new Edge(rez.center, parc.prev.focus, parc.focus);
411             parc.prev.edge1 = parc.edge0;
412             parc.edge1 = new Edge(rez.center, parc.focus, parc.next.focus);
413             parc.next.edge0 = parc.edge1;
414
415             //save information for removing colinear edges
416             rez.center.mech = true;
417             rez.center.mech0 = parc.edge0;
418         }
419     }
420 }

```

```

421         rez.center.mech1 = parc.edge1;
422         mechs.add(rez.center);
423
424         // Check for new circle events around the new arc:
425         checkCircleEvent(parc, focus.x);
426         checkCircleEvent(parc.prev, focus.x);
427         checkCircleEvent(parc.next, focus.x);
428
429         return;
430     }
431
432     //proceed to next arc
433     parc = parc.next;
434 }
435
436 // Special case: If p never intersects an arc, append it to the list.
437 parc = root;
438 while (parc.next != null)
439     parc = parc.next; // Find the last node.
440 parc.next = new Arc(focus);
441 parc.next.prev = parc;
442 Point start = new Point(0, (parc.next.focus.y + parc.focus.y) / 2);
443 parc.next.edge0 = new Edge(start, parc.focus, parc.next.focus);
444 parc.edge1 = parc.next.edge0;
445
446 //focus.verts.addFirst(start);
447
448 }
449
450 void checkCircleEvent(Arc parc, double xpos) {
451     // Invalidate any old event.
452     if ((parc.event != null) && (parc.event.xpos != xpos))
453         parc.event.valid = false;
454     parc.event = null;
455
456     if ((parc.prev == null) || (parc.next == null)) return;
457
458     CircleResultPack result = circle(parc.prev.focus, parc.focus, parc.next.focus);
459     if (result.valid && result.rightmostX > xpos) {
460         // Create new event.
461         parc.event = new CircleEvent(result.rightmostX, result.center, parc);
462         events.offer(parc.event);
463     }
464 }
465
466 // Find the rightmost point on the circle through a,b,c.
467 CircleResultPack circle(Point a, Point b, Point c) {
468     CircleResultPack result = new CircleResultPack();
469
470     // Check that bc is a "right turn" from ab.
471     if ((b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y) > 0) {
472         result.valid = false;
473         return result;
474     }
475
476     // Algorithm from O'Rourke 2ed p. 189.
477     double A = b.x - a.x;
478     double B = b.y - a.y;
479     double C = c.x - a.x;
480     double D = c.y - a.y;
481     double E = A * (a.x + b.x) + B * (a.y + b.y);
482     double F = C * (a.x + c.x) + D * (a.y + c.y);
483     double G = 2 * (A * (c.y - b.y) - B * (c.x - b.x));
484
485     if (G == 0) { // Points are co-linear.
486         result.valid = false;
487         return result;
488     }
489
490     // centerpoint of the circle.
491     Point o = new Point((D * E - B * F) / G, (A * F - C * E) / G);
492     result.center = o;
493
494     // o.x plus radius equals max x coordinate.
495     result.rightmostX = o.x + Math.sqrt(Math.pow(a.x - o.x, 2.0) + Math.pow(a.y - o.y, 2.0));
496
497     result.valid = true;
498     return result;
499 }
500
501 // Will a new parabola at point p intersect with arc i?
502 CircleResultPack intersect(Point p, Arc i) {
503     CircleResultPack res = new CircleResultPack();
504     res.valid = false;
505     if (i.focus.x == p.x) return res;
506
507     double a = 0.0;
508     double b = 0.0;
509     if (i.prev != null) // Get the intersection of i->prev, i.
510         a = intersection(i.prev.focus, i.focus, p.x).y;
511     if (i.next != null) // Get the intersection of i->next, i.
512         b = intersection(i.focus, i.next.focus, p.x).y;
513
514     if ((i.prev == null || a <= p.y) && (i.next == null || p.y <= b)) {
515         res.center = new Point(xmin, p.y);
516
517         // Plug it back into the parabola equation to get the x coordinate
518         res.center.x = (i.focus.x * i.focus.x + (i.focus.y - res.center.y) * (i.focus.y - res.center.y) - p.x *
519             res.valid = true;
520         return res;
521     }
522     return res;
523 }
524
525 // Where do two parabolas intersect?
526 Point intersection(Point p0, Point p1, double l) {
527     Point res = new Point(xmin, ymin);
528     Point p = p0;
529
530     if (p0.x == p1.x) {
531         res.y = (p0.y + p1.y) / 2;
532     } else if (p1.x == l) {
533         res.y = p1.y;
534     } else if (p0.x == l) {
535         res.y = p0.y;
536     } else {
537         p = p1;
538         // Use the quadratic formula.
539         double z0 = 2 * (p0.x - l);
540         double z1 = 2 * (p1.x - l);
541
542         double a = 1 / z0 - 1 / z1;
543         double b = -2 * (p0.y / z0 - p1.y / z1);
544         double c = (p0.y * p0.y + p0.x * p0.x - l * l) / z0 - (p1.y * p1.y + p1.x * p1.x - l * l) / z1;
545
546         res.y = (-b - Math.sqrt((b * b - 4 * a * c))) / (2 * a);
547
548         // Plug back into one of the parabola equations.
549         res.x = (p.x * p.x + (p.y - res.y) * (p.y - res.y) - l * l) / (2 * p.x - 2 * l);
550         return res;
551     }
552 }
553
554 void finishEdges() {
555     // Advance the sweep line so no parabolas can cross the bounding box.
556     double l = width * 2 + height;
557
558     // Extend each remaining segment to the new parabola intersections.
559
560     Arc i = root;
561     while (i != null) {
562         if (i.edge1 != null) {
563             Point fp = intersection(i.focus, i.next.focus, l * 2);
564             i.edge1.finish(fp);
565         }
566         i = i.next;
567     }
568 }
569
570 void polyFinish() {
571     //clean up mechs
572     //These are single lines formed by 3 colinear points
573     //they are a normal result of fortune's algorithm
574     //We only want 2 points
575     Point next;
576 }

```

```

Point unpoint;
ListIterator<Point> mek = mechs.listIterator(0);
while (mek.hasNext()) {
    unpoint = mek.next();
    unpoint.mech0.start = unpoint.mech1.end;
    unpoint.mech1.done = false;
}

//create a TreeSet<APoint> for every site
//APoints will be ordered by thier angle from the site.
Point pk;
ListIterator<Point> eek = pts.listIterator(0);
while (eek.hasNext()) {
    pk = eek.next();
    pk.sortSet = new TreeSet<Point>();
    pk.sverts = new TreeSet<APoint>();
    pk.neighbors = new TreeSet<Point>();
}

//define the bounding box
Point[] xing = new Point[4];

//the corners of the bounding box
topleft = new Point(xmin,ymin);
topright = new Point(xmax,ymin);
bottomleft = new Point(xmin,ymax);
bottomright = new Point(xmax,ymax);

//four sorted sets of Points that lie along each side of the bounding box
TreeSet<Point> sides = new TreeSet<Point>();
sides[0] = new TreeSet<Point>(); //top
sides[1] = new TreeSet<Point>(); //right
sides[2] = new TreeSet<Point>(); //bottom
sides[3] = new TreeSet<Point>(); //left

//add the corners to the lists of sidepoints
/*
sides[0].add(topleft);
sides[0].add(topright);
sides[1].add(topright);
sides[1].add(bottomright);
sides[2].add(bottomright);
sides[2].add(bottomleft);
sides[3].add(bottomleft);
sides[3].add(topleft);
*/

//the edge in question
Edge e;
ListIterator<Edge> j;

/*
 * loop over the edges
 * trim edges that intersect the bounding box,
 * and add edges between those points of intersection,
 * that close all the polygons that were opened by the trimming.
 */
j = output.listIterator(0);
while (j.hasNext()) {
    e = j.next();

    if (e.done){
        //check if the edge intersects with the bounds
        xing[0] = lineIntersect( topleft, topright, e.start, e.end );
        xing[1] = lineIntersect( topright, bottomright, e.start, e.end );
        xing[2] = lineIntersect( bottomright, bottomleft, e.start, e.end );
        xing[3] = lineIntersect( bottomleft, topleft, e.start, e.end );

        for (int ii=0; ii<4; ii++){
            //top, right, bottom, left
            //inside function catches double precision mistakes
            if (xing[ii] != null && inside(xing[ii])) {
                // e intersects side ii
                if (ii==0)
                    xing[ii].y = ymin;
                else if (ii==1)
                    xing[ii].x = xmax;
                else if (ii==2)
                    xing[ii].y = ymax;
                else if (ii==3)
                    xing[ii].x = xmin;

                sides[ii].add(xing[ii]);
                xing[ii].assoc = e;

                //shorten e
                if (outside(e.start)){
                    e.start = xing[ii];
                }
                else {
                    e.end = xing[ii];
                }
            }
        }
    }
}

/*
 * loop over side points
 * all intersections have been found, turn lists of side points into edges with proper
 * site associations
 */
Iterator<Point> jj;
Point sp,osp; //sidepoint, old sidepoint

for (int ii=0; ii<4; ii++){ // for each side
    if (sides[ii].size() >= 2){ // if there was anything more than just the corner points
        jj = sides[ii].iterator();
        sp = jj.next();

        while (jj.hasNext()){
            osp = sp;
            sp = jj.next();

            //must determine which site this edge should border
            //will be the site that both edges connected to
            //sp and osp have in common
            //note this does not handle corner cases
            Point commonSite = null;
            if (osp.assoc != null && sp.assoc != null){
                if (osp.assoc.siteA == sp.assoc.siteA || osp.assoc.siteA == sp.assoc.siteB) {
                    commonSite = osp.assoc.siteA;
                } else if (osp.assoc.siteB == sp.assoc.siteA || osp.assoc.siteB == sp.assoc.siteB) {
                    commonSite = osp.assoc.siteB;
                }
            }

            Edge xo = new Edge(osp, commonSite, null);
            xo.finish(sp);
        }
    }

    //associate corners with sites
    Point commonSite = null;
    Point corner = null;
    // standing at the corner, looking in, sp will be the sidepoint on your right and osp will be to your left.
    osp = null;
    sp = null;

    for (int ii=0; ii<4; ii++) {
        if (ii == 0){
            corner = topleft;
            sp = (Point)(sides[3].first());
            osp = (Point)(sides[0].first());
        }
        else if (ii == 1){
            corner = topright;
            sp = (Point)(sides[0].last());
            osp = (Point)(sides[1].first());
        }
        else if (ii == 2){
            corner = bottomright;
            sp = (Point)(sides[1].last());
            osp = (Point)(sides[2].first());
        }
        else if (ii == 3){
            corner = bottomleft;
            sp = (Point)(sides[2].last());
            osp = (Point)(sides[3].first());
        }
    }
}

```

```

734     corner = bottomright;
735     sp = (Point)(sides[1].last());
736     osp = (Point)(sides[2].last());
737     } else if (ii == 3){
738         corner = bottomleft;
739         sp = (Point)(sides[2].first());
740         osp = (Point)(sides[3].last());
741     }
742
743     if (osp != null && sp != null) {
744         if ( ( osp.assoc.siteA == sp.assoc.siteA || osp.assoc.siteA == sp.assoc.siteB) && osp.assoc.siteA != null
745             commonSite = osp.assoc.siteA;
746         } else if ( ( osp.assoc.siteB == sp.assoc.siteA || osp.assoc.siteB == sp.assoc.siteB) && osp.assoc.site
747             commonSite = osp.assoc.siteB;
748         }
749
750         Edge xo = new Edge( osp, commonSite, null );
751         xo.finish(corner);
752         Edge po = new Edge( sp, commonSite, null );
753         po.finish(corner);
754     }
755 }
756
757
758
759
760 /*
761  * Loop over edges again
762  * add the endpoints of the edge to a sorted set of points,
763  * the Points are sorted based on thier angle from the site
764  * associated with both of the sites that are next to this edge
765  * there are only ever two, but for the boundary cases, one of them will be null,
766  * ignore it.
767  */
768
769 j = output.listIterator(0);
770 while (j.hasNext()) {
771     e = j.next();
772     if (e.done){
773         //add the edge's endpoints to the sorted set of verts for the sites on each side.
774         if (e.siteA != null){
775             if (loutside(e.start))
776                 e.siteA.sortSet.add( e.start );
777             if (loutside(e.end))
778                 e.siteA.sortSet.add( e.end );
779         }
780         if (e.siteB != null){
781             if (loutside(e.start))
782                 e.siteB.sortSet.add( e.start );
783             if (loutside(e.end))
784                 e.siteB.sortSet.add( e.end );
785         }
786         if ((e.siteA != null) && (e.siteB != null)){
787
788             Coordinate[] coords = new Coordinate[2];
789             coords[0] = new Coordinate(e.start.x, e.start.y);
790             coords[1] = new Coordinate(e.end.x, e.end.y);
791             LineString ls = geometryFactory.createLineString(coords);
792
793             //boolean isCover = zonePoly.covers(ls);
794             boolean isIntersect = zonePoly.intersects(ls);
795             //geometryFactory
796
797             if (isIntersect /*isCover*/){
798                 e.siteA.neighbors.add(e.siteB);
799                 e.siteB.neighbors.add(e.siteA);
800             }
801         }
802     }
803 }
804
805 }
806
807
808 /*
809  * Loop over the sites
810  * for each site, take the sorted set of Points (in acending x value)
811  * and add them all to the sorted set of APoints (in acending angle from site)
812  */
813 Iterator<Point> uu;
814 Point eko;
815 listIterator<Point> oo = pts.listIterator(0);
816 Point site;
817 while (oo.hasNext()) {
818     site = oo.next();
819
820     if (site.sortSet.size() >= 3){
821         uu = site.sortSet.iterator();
822         while (uu.hasNext()){
823             eko = uu.next();
824             site.sverts.add( new APoint(eko, radFromPoints(site,eko) ) );
825         }
826     }
827 }
828
829 }
830
831
832 boolean outside(Point q){
833     //is a Point outside the bounding box?
834     return ( (q.x < xmin) ||
835             (q.x > xmax) ||
836             (q.y < ymin) ||
837             (q.y > ymax) );
838 }
839
840 boolean inside(Point q){
841     //is a Point outside the bounding box?
842     return ( (q.x >= xmin - 1) &&
843             (q.x <= xmax + 1) &&
844             (q.y >= ymin - 1) &&
845             (q.y <= ymax + 1) );
846 }
847
848 double radFromPoints(Point A, Point B){
849     double dx = B.x-A.x;
850     double dy = B.y-A.y;
851     double rad = Math.atan(dy/dx);
852     if(dx<0) rad+=Math.PI;
853     return rad;
854 }
855
856 class Point implements Comparable<Point> {
857
858     public double x, y;
859     public int key;
860
861     public TreeSet<Point> sortSet;
862     public TreeSet<APoint> sverts;
863     public TreeSet<Point> neighbors;
864     public Edge assoc;
865
866     public boolean mech = false;
867     public Edge mech0, mech1;
868
869     public int fillcolor;
870
871     public Point(double X, double Y) {
872         x = X;
873         y = Y;
874     }
875
876     public int compareTo(Point foo) {
877         //return zero only if this is the same object.
878         if (this == foo) {
879             return 0;
880         } else {
881             int rx = ((Double) this.x).compareTo((Double) foo.x);
882             if (rx != 0){
883                 return rx;
884             } else {
885                 int ry = ((Double) this.y).compareTo((Double) foo.y);
886                 if (ry != 0){
887                     return ry;
888                 } else {
889                     return 1;
890                 }
891             }
892         }
893     }
894 }

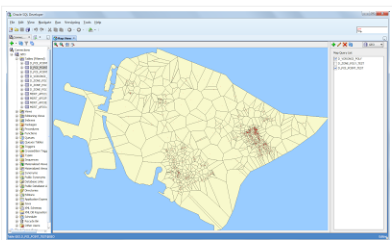
```

```

890     }
891   }
892 }
893 }
894 }
895 }
896
897 class APoint implements Comparable<APoint>{
898
899   public double x, y;
900   public double angle;
901
902   public APoint(Point p, double ang){
903     x = p.x;
904     y = p.y;
905     angle = ang;
906   }
907
908   public int compareTo(APoint foo) {
909     //return zero only if this is the same object.
910     if (this == foo) {
911       return 0;
912     } else {
913       return ((Double) this.angle).compareTo((Double) foo.angle);
914     }
915   }
916 }
917
918 class CircleEvent implements Comparable<CircleEvent> {
919
920   public double xpos;
921   public Point p;
922   public Arc a;
923   public boolean valid;
924
925   public CircleEvent(double X, Point P, Arc A) {
926     xpos = X;
927     a = A;
928     p = P;
929     valid = true;
930   }
931
932   public int compareTo(CircleEvent foo) {
933     return ((Double) this.xpos).compareTo((Double) foo.xpos);
934   }
935 }
936
937 class Edge {
938
939   public Point start, end;
940   public boolean done;
941   public Point siteA, siteB;
942
943   public Edge(Point p, Point sitea, Point siteb) {
944     start = p;
945     end = new Point(0, 0);
946     done = false;
947     siteA = sitea;
948     siteB = siteb;
949     output.add(this);
950   }
951
952   public void finish(Point p) {
953     if (done) {
954       return;
955     }
956     end = p;
957     done = true;
958   }
959 }
960
961 class Arc {
962   //parabolic arc is the set of points eqadistant from a focus point and the beach line
963
964   public Point focus;
965   //these object exsist in a linked list
966   public Arc next, prev;
967   //
968   public CircleEvent event;
969   //
970   public Edge edge0, edge1;
971
972   public Arc(Point p) {
973     focus = p;
974     next = null;
975     prev = null;
976     event = null;
977     edge0 = null;
978     edge1 = null;
979   }
980 }
981
982 class CircleResultPack {
983   public boolean valid;
984   public Point center;
985   public double rightmostX;
986 }
987
988 Point midpoint(Point a, Point b){
989   return new Point( (a.x+b.x)/2, (a.y+b.y)/2 );
990 }
991
992 Point lineIntersect(Point A, Point B, Point C, Point D){
993   /*
994   * if line segment AB intersects line segment CD,
995   * return the point of intersection
996   * else return null
997   */
998   double ex = B.x - A.x;
999   double ey = B.y - A.y;
1000   double fx = D.x - C.x;
1001   double fy = D.y - C.y;
1002   double h = ( (A.x-C.x) * (-ey) + (A.y-C.y) * ex ) / (fx * (-ey) + fy * ex);
1003   if (0 <= h && h <= 1){
1004     return new Point( C.x + h * fx,
1005       C.y + h * fy );
1006   } else return null;
1007 }
1008 }

```

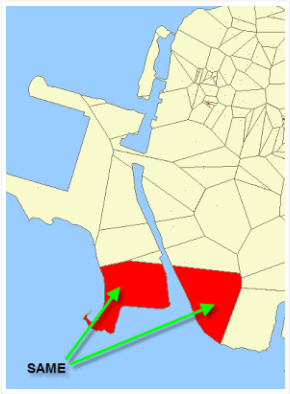
10) После запуска java-программы проверим наличие данных в таблице D_VORONOI_POLY с помощью sqldeveloper (рекомендую скачать его и использовать для работы со Spatial-объектами БД Oracle)



11) Дальше вы сами вправе решать, что делать с получившимися полигонами... склеить их, или использовать в своих эвристических алгоритмах...

Но следует помнить о ряде ограничений данного решения:

- Диаграмма строится по точкам внутри замкнутого полигона, если полигон содержит анклавы - ситуацию придется как-то обрабатывать;
- Вам желательно самим предусмотреть обработку точек-дубликатов (когда несколько РАЗНЫХ домов имеют одинаковые координаты);
- Если ограничивающий полигон не является выпуклым многоугольником, то возможны "некрасивые ситуации" (следует помнить, что сама диаграмма строится внутри MBR - "минимального охватывающего точки прямоугольник" - и уже затем от каждой ячейки Вороного "отрезаются кусочки" за границами охватывающего полигона).



На этом все, надеюсь кому-то это поможет!

Автор: Sergey Sheremeta на 3:31
Ярлыки: карты, map, Spatial, Voronoi

Комментариев нет:

Отправить комментарий

Введите комментарий...

Подпись комментария:

Andrey Alisov

Выйти

Публикация

Просмотр

☐ Оповещать