70 / 686

# Chapter 6

**使用PostGIS几何对象：构建应用**

**6.1**

**使用MapServer**

Minnesota MapServer 是一个web地图服务器，支持OpenGIS Web Mapping Server 规范.

·MapServer主页 [http://mapserver.org.](http://mapserver.org)

·OpenGIS Web Map Speciﬁcation 规范 [http://www.opengeospatial.org/standards/wms.](http://www.opengeospatial.org/standards/wms)

**6.1.1**

**基础使用**

想知道怎样将PostGIS 和 MapServer搭配使用，你需要知道怎样配置MapServe，这超出了本文档的内容。本节主要聚焦PostGIS的问题和和怎样配置PostGIS。

要将PostGIS和MapServer搭配使用，你需要：

· PostGIS 0.6版本或更高版本

·MapServer 3.5版本或更高版本

MapServer 和其他PostgreSQL客户端一样，使用libpq接口获取 PostGIS/PostgreSQL 数据。这意味着MapServer可以安装在任何可用于PostGIS 服务器网络连通的服务器上面，并使用PostGIS做数据源。两个系统之间连接速度越快越好。

1. 编译并安装MapServer，编译参数根据你的选择，但需要带上"--with-postgis" 配置选项

2. 在你的MapServer map文件中，添加一个PostGIS图层，例如：

LAYER

CONNECTIONTYPE postgis

NAME "widehighways"

# Connect to a remote spatial database

CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"

PROCESSING "CLOSE\_CONNECTION=DEFER"

# Get the lines from the ‘geom’ column of the ‘roads’ table

DATA "geom from roads using srid=4326 using unique gid"

STATUS ON

TYPE LINE

# Of the lines in the extents, only render the wide highways

FILTER "type = ‘highway’ and numlanes >= 4"

CLASS

# Make the superhighways brighter and 2 pixels wide

71 / 686

EXPRESSION ([numlanes] >= 6)

STYLE

COLOR 255 22 22

WIDTH 2

END

END

CLASS

# All the rest are darker and only 1 pixel wide

EXPRESSION ([numlanes] < 6)

STYLE

COLOR 205 92 82

END

END

END

在上面的例子中，关于PostGIS的具体MapServer指令语义如下：

CONNECTIONTYPE ：对于PostGIS图层，该值总是”postgis”

CONNECTION 数据库连接是由一个连接字符串来控制，该字符串包括一系列标准格式的键值对：

user=<username> password=<password> dbname=<username> hostname=<server> port=<5432>

一个空连接字符串也是有效的，因为可以被省略掉。最少你要提供数据库名称和用户名

DATA 该参数的形式是 "<geocolumn> from <tablename> using srid=<srid> using unique <primary key>"，column列是空间类型的数据列，该列的数据源会在地图中被渲染，SRID是几何列的SRID值，primary key是表的primary key（或者唯一索引列）。你可以省略 "using srid" 和 "using unique" 子句， MapServer 如果能的话可以自动确认正确的值，但是会在MapServer上对每一个绘图耗费一些额外的查询该值。

PROCESSING 值为“CLOSE\_CONNECTION=DEFER”，如果你有多个图层需要重复使用该数据库连接，不用关闭他们。这可以提高加载速度。参考MapServer PostGIS Performance Tips 获取更多信息。

FILTER 该过滤条件必须是一个有效的SQL字符串，与正常的SQL里面的WHERE子句相对应。因此，举个例子，要渲染车道多于6条及6条以上的路，使用过滤条件"num\_lanes >= 6"。

3.在你的空间数据库中，确保你有你要绘图的字段有空间索引（GiST索引）

CREATE INDEX [indexname] ON [tablename] USING GIST ( [geometrycolumn] );

4.如果你要使用MapServer来查询你的图层，你同样需要在DATA表达式中使用"using unique" 子句

当查询的时候，MapServer 需要对于每一行记录有一个唯一值来标识。MapServer的PostGIS模块会使用这些你指定的唯一值。表的primary key是最好的唯一值标识字段

**6.1.2**

**常见问题**

1. 当我在map文件中配置EXPRESSION 参数时候，从返回记录上看，过滤条件从来没有返回true值，即便我确信记录一定存在表中。

答：不像shape文件， EXPRESSION 参数中需要使用PostGIS 字段名称的小写形式。

EXPRESSION ([numlanes] >= 6)

2. 我对shape格式文件使用的参数FILTER对于我的PostGIS表的同样数据不OK，什么原因呢？

答：不同于shape格式文件，PostGIS图层使用SQL语法（这些条件被添加到SQL语句中用于在MapServer中进行图形渲染）的过滤条件

72 / 686

FILTER "type = ‘highway’ and numlanes >= 4"

3. 我的PostGIS 图层渲染要比shape格式的文件数据渲染慢得多，这正常吗？

答：总体来看，你要在一个图上面所渲染的特征越多，PostGIS的数据比shape格式文件渲染要慢的可能性越大。对于那些渲染特征相对要少的地图（100秒左右），PostGIS通常会更快。对于那些渲染特征很多的图（渲染时间1000s左右甚至更高），PostGIS总是要慢很多。如果你正在寻找图形渲染慢的真正问题，很可能是因为你没有在你的表上面创建空间索引。

postgis# CREATE INDEX geotable\_gix ON geotable USING GIST ( geocolumn );

postgis# VACUUM ANALYZE;

4. 我的PostGIS图层渲染得还不错，但查询非常慢，上面问题？

为了让查询很快，你的空间表上面必须要有一个唯一的key，同时在这个key上面要有一个索引。你可以在map文件的DATA参数上面使用USING UNIQUE子句指定唯一的key：

DATA "geom FROM geotable USING UNIQUE gid"

5. 我可以在MapServer上面使用geography类型列（PostGIS 1.5新的数据类型）作为数据源吗？

答：当然可以！ MapServer 可以像geometry数据类型一样理解geography类型列，但是总是会将SRID值认为是4326.只需要在你的map文件的DATA参数中加上"using srid=4326" 值。所有的MapServer功能都会像geometry类型列一样使用

DATA "geog FROM geogtable USING SRID=4326 USING UNIQUE gid"

**6.1.3**

**高级用法**

在map文件中使用伪SQL语句添加一些信息是为了帮助MapServer理解一些复杂查询结构。更具体地说，更具体地说，无论在MapServer中用视图还是用子查询当做数据源（即map文件中的DATA参数的FROM子句），对MapServer来说是很难自动确定每一行的唯一记录标识和SRID的。USING子句可以给MapServer提供两方面信息如下：

DATA "geom FROM (

SELECT

table1.geom AS geom,

table1.gid AS gid,

table2.data AS data

FROM table1

LEFT JOIN table2

ON table1.id = table2.id

) AS new\_table USING UNIQUE gid USING SRID=4326"

USING UNIQUE <uniqueid> ：MapServer在做地图查询时候，对于每一行需要一个标识该行唯一的字段。正常情况下，会使用表的primary key当做唯一标识。然而视图和子查询不能辨别那个是行的唯一标识。如果你想使用MapServer的查询功能，你需要保证你的视图或者子查询包含一个唯一标识列，然后使用USING UNIQUE 来声明。比如，你可以在查询中包含primary key列或者其他任何可用保证该记录唯一的列。

**注意**

"查询地图 "是指点击地图上某个位置，获取该位置上的地图特征信息。不要把“地图查询”和DATA参数中定义的SQL查询混淆了！

USING SRID=<srid> PostGIS 需要知道几何数据列使用的是哪种空间参考系，以便返回正确的数据给MapServer。正常情况下，可以在PostGIS数据库的系统视图"geometry\_columns" 中找到该SRID信息，然而对于临时创建的子查询和视图是不能找到这些信息的。因此允许在DATA参数中指定USING SRID= 选项



73 / 686

**6.1.4**

**样例**

让我们从一个简单的例子开始来验证我们的方法。考虑下面的MapServer图层定义：

LAYER

CONNECTIONTYPE postgis

NAME "roads"

CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"

DATA "geom from roads"

STATUS ON

TYPE LINE

CLASS

STYLE

COLOR 0 0 0

END

END

END

这个图层会展示表roads的所有道路图，以黑线表示。现在我们想在至少1:100000比例下，只显示高速公路——下面两个图层会达到这种效果

LAYER

CONNECTIONTYPE postgis

CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"

PROCESSING "CLOSE\_CONNECTION=DEFER"

DATA "geom from roads"

MINSCALE 100000

STATUS ON

TYPE LINE

FILTER "road\_type = ‘highway’"

CLASS

COLOR 0 0 0

END

END

LAYER

CONNECTIONTYPE postgis

CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"

PROCESSING "CLOSE\_CONNECTION=DEFER"

DATA "geom from roads"

MAXSCALE 100000

STATUS ON

TYPE LINE

CLASSITEM road\_type

CLASS

EXPRESSION "highway"

STYLE

WIDTH 2

COLOR 255 0 0

END

END

CLASS

STYLE

COLOR 0 0 0

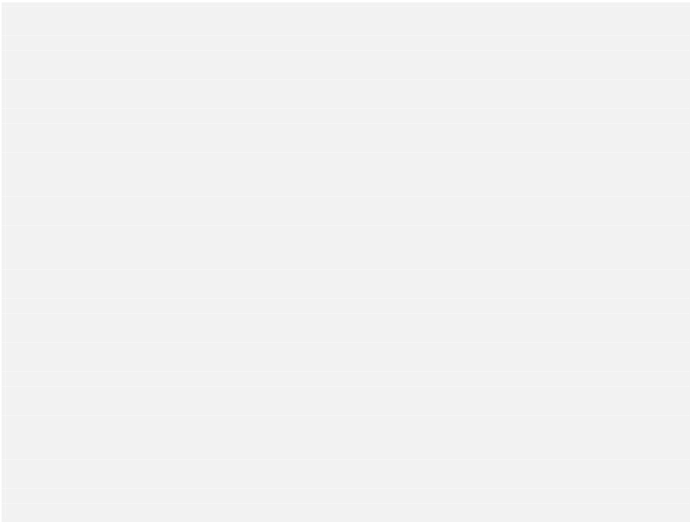
END

END

END

第一个图层是比例高于1:100000的情形，只以黑线展示高速路。

FILTER 参数限制只展示高速公路



74 / 686

第二个图层用于比例小于1:100000的情况，双击红线的时候会展示高速公路，而其他的道路还是用黑线表示。

现在我们已经使用MapServer功能做了一些有趣的事情，但我们DATA 参数里面的SQL语句是比较简单。假设道路的名称是存储在其他表中（因为别的原因），这样的话我们需要做一个join关联来得到它以便标记我们的道路。

LAYER

CONNECTIONTYPE postgis

CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"

DATA "geom FROM (SELECT roads.gid AS gid, roads.geom AS geom,

road\_names.name as name FROM roads LEFT JOIN road\_names ON

roads.road\_name\_id = road\_names.road\_name\_id)

AS named\_roads USING UNIQUE gid USING SRID=4326"

MAXSCALE 20000

STATUS ON

TYPE ANNOTATION

LABELITEM name

CLASS

LABEL

ANGLE auto

SIZE 8

COLOR 0 192 0

TYPE truetype

FONT arial

END

END

END

这个图层在地图比例低于1:20000时候，所有的道路会显示成绿色的。该样例也展示了怎样在DATA参数定义中使用SQL join的功能。

**6.2**

**Java 客户端 (JDBC)**

Java客户端要么通过直接的文本描述对象的方式要么通过打包了PostGIS jar包的JDBC驱动来获取PostgreSQL数据库中的几何数据。为了使用JDBC扩展，postgis.jar文件必须和JDBC驱动文件postgresql.jar一样在CLASSPATH变量所对应的目录中

import

import

import

import

java.sql.\*;

java.util.\*;

java.lang.\*;

org.postgis.\*;

public class JavaGIS {

public static void main(String[] args) {

java.sql.Connection conn;

try {

/\*

\* Load the JDBC driver and establish a connection.

\*/

Class.forName("org.postgresql.Driver");

String url = "jdbc:postgresql://localhost:5432/database";

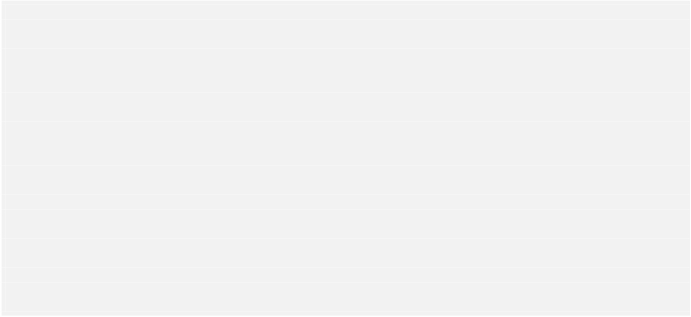
conn = DriverManager.getConnection(url, "postgres", "");

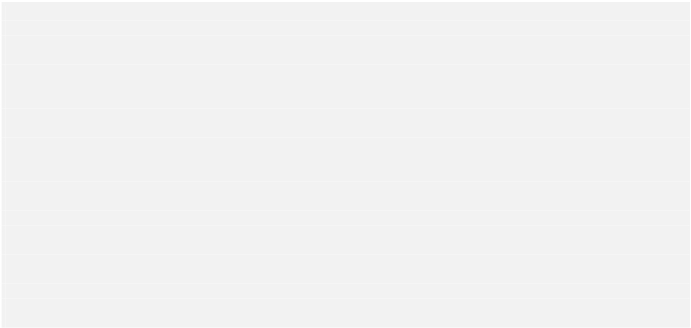
/\*

\* Add the geometry types to the connection. 注意 that you

\* must cast the connection to the pgsql-specific connection

\* implementation before calling the addDataType() method.





75 / 686

\*/

((org.postgresql.PGConnection)conn).addDataType("geometry",Class.forName("org.postgis. ←

PGgeometry"));

((org.postgresql.PGConnection)conn).addDataType("box3d",Class.forName("org.postgis. ←

PGbox3d"));

/\*

\* Create a statement and execute a select query.

\*/

Statement s = conn.createStatement();

ResultSet r = s.executeQuery("select geom,id from geomtable");

while( r.next() ) {

/\*

\* Retrieve the geometry as an object then cast it to the geometry type.

\* Print things out.

\*/

PGgeometry geom = (PGgeometry)r.getObject(1);

int id = r.getInt(2);

System.out.println("Row " + id + ":");

System.out.println(geom.toString());

}

s.close();

conn.close();

}

catch( Exception e ) {

e.printStackTrace();

}

}

}

变量"PGgeometry" 对象是一个包含具体的拓扑几何对象（抽象类Geometry的之类）的包装对象。拓扑几何对象类型包括: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon类型

PGgeometry geom = (PGgeometry)r.getObject(1);

if( geom.getType() == Geometry.POLYGON ) {

Polygon pl = (Polygon)geom.getGeometry();

for( int r = 0; r < pl.numRings(); r++) {

LinearRing rng = pl.getRing(r);

System.out.println("Ring: " + r);

for( int p = 0; p < rng.numPoints(); p++ ) {

Point pt = rng.getPoint(p);

System.out.println("Point: " + p);

System.out.println(pt.toString());

}

}

}

JavaDoc工具为不同的获取几何对象的Java函数提供了API参考

**6.3**

...

**6.3.1**

...

**6.3.2**

...

**C 客户端 (libpq)**

译者注：下面两小节均无细节文章，标题是PostgreSQL的两种游标

**Text Cursors**

**Binary Cursors**

