PostGIS 2.2.0dev 开发手册

58 / 686

**Chapter 5**

**栅格数据管理，查询和应用**

**5.1**

**加载和创建栅格**

大多数使用场景中，用户通常需要使用栅格数据加载程序raster2pgsql来把已有的栅格文件来创建PostGIS栅格

**5.1.1**

**使用raster2pgsql来加载栅格**

加载程序raster2pgsql是一个可执行的栅格加载工具，它把支持GDAL格式的栅格文件转换成SQL化的文件，以便将其加载到PostGIS栅格表中。 It is capable of loading folders of raster ﬁles as well as creating overviews of rasters.

由于raster2pgsql大多数是在安装PostGIS时候编译的（除非你自己编译自己的GDAL库），raster2pgsql支持的栅格类型将和所依赖的GDAL库支持的类型是一样的。想要获取你的给定版本的raster2pgsql所支持的栅格类型，请使用-G参数。如果两边库一样的话，输出结果应该和ST\_GDALDrivers输出结果一样。

**注意**

这个工具老版本是用Python写的脚本。新的可执行程序已经替换了Python脚本。如果你依然想看Python脚本的样例，你可以在这里查找。GDAL PostGIS Raster Driver Usage(<http://trac.osgeo.org/gdal/wiki/frmts_wtkraster.html).请注意Python脚本方式的raster2pgsql对于未来版本的PostGIS>栅格可能无法使用，并且也将不再支持。

**注意**

当使用函数ST\_CreateOverview从一个已经对齐的栅格并指定参数factor因子创建overview时候，创建出的overview可以是不对齐的。align. 参考 http://trac.osgeo.org/postgis/ticket/1764 获取不对齐的overview样例

使用样例:

raster2pgsql raster\_options\_go\_here raster\_file someschema.sometable > out.sql

-? 该参数展示帮助信息。如果执行raster2pgsql程序时候，不传递任何参数，也会展示帮助信息。

-G 打印支持的栅格格式

(c|a|d|p) 下面这些选项互斥:



PostGIS 2.2.0dev 开发手册

59 / 686

-c 创建新表，并把栅格数据导入到表中，这是默认模式

-a 追加栅格数据到一个已经存在的表

-d 先drop掉表，然后创建新表，并把栅格数据导入到表中

-p 准备模式，只创建表，不导入数据.

栅格处理过程：把适当的约束条件注册（写入）栅格系统表中。

-C 用于栅格约束注册——比如栅格的srid, pixelsize 元数据信息等等，保证栅格在视图raster\_columns能够合适地注册。

-x 让设置最大边界约束失效，只有在-C参数同时使用时候才能使用这个参数。

-r 为规则的块设置约束（空间唯一性约束和覆盖瓦片）。只有在同时使用-C 参数时，才能使用这个参数。

栅格处理参数：用于操作输入栅格数据集的可选参数。

-s 表示<SRID>：指定输出栅格使用指定的SRID。如果没有提供或者提供值为0，程序会检查栅格的元数据（比如栅格文件）以便决定一个合适的SRID。

-b 从栅格要抽取的波段位置（下标从1开始）。想要抽取不止1个波段的话，使用逗号，分隔波段数字。如果没有指定，所有栅格的波段都会被抽取出来。

-t TILE\_SIZE：瓦片大小。把每一个表的行表示的栅格切割成瓦片。TILE\_SIZE被表示成WIDTHxHEIGHT的形式，或者设置成值为“auto”来让加载程序使用第一个栅格参数计算合适的瓦片大小，然后把这个瓦片大小应用于所有的栅格

-R, --register 把栅格以文件数据形式注册（栅格数据保存在数据库之外的文件系统中）。只有栅格的元数据和文件路径保存在数据库中（没有保存像素的数据）

-l **OVERVIEW\_FACTOR** ：创建栅格的概览信息。如果factor值不只一个，使用逗号，进行分隔。创建的概览数据存储在数据库中，并且不受参数-R影响。注意你生成的SQL文件会包含存储数据的主表和概览表。

-N **NODATA** ：指定没有NODATA值的波段的NODATA值。

用于操作数据库对象的可选参数

-q ：把PostgreSQL的id用引号包起来。

-f COLUMN :指定目标的栅格列名，默认是‘rast’。

-F ：创建一个以文件名为列名的列

-I ：在栅格列上创建一个GiST索引

-M Vacuum analyze栅格表

-T **tablespace** ：指定新表的表空间。注意如果索引（包括primary key）依然会使用默认的表空，除非使用-X参数标识。

-X **tablespace** ：指定表的索引表空间。如果使用了参数-I，那么这个参数可以应用于primary key和空间索引。

-Y ：使用copy语句而不是insert 语句。

-e ：单独地执行每一条语句，不使用一个大的事务执行所有语句。

-E ENDIAN ：控制栅格输出时候生成的二进制的字节顺序，如果值为0，表示取XDR，如果值为1，表示取NDR（默认取值）。现在我们只支持NDR了。

-V **version：**指定输出版本的格式，默认是0，当前值支持0。

一个使用raster2pgsql的用于创建一个sql输入文件，并生成一个100x100 方格的样例如下：

**注意**

你可以省略表的schema名，比如demelevation而不是public.demelevation ，这样栅格表就会创建在数据库默认的schema下面



PostGIS 2.2.0dev 开发手册

60 / 686

raster2pgsql -s 4326 -I -C -M \*.tif -F -t 100x100 public.demelevation > elev.sql

psql -d gisdb -f elev.sql

可以使用UNIX管道把转换和加载一步完成：

raster2pgsql -s 4326 -I -C -M \*.tif -F -t 100x100 public.demelevation | psql -d gisdb

这样就把栅格数据Massachusetts州的平面瓦片数据加载一个名为aerial的schema中，然后创建一个完整视图，然后使用copy语句把数据直接导入到数据库中。参数-e不要求所有的插入语句都在一个事务中（这个参数的好处是可以让你想立即在表中看到部分数据，因为分步提交可以不等待数据完全插入完成）。程序会把栅格数据分成128x128的像素瓦片，然后加上栅格约束。这里使用copy而不是insert方式。加上参数-F会包含一个命名为从瓦片提取数据的文件名的字段。

raster2pgsql -I -C -e -Y -F -s 26986 -t 128x128

-l 2,4 bostonaerials2008/\*.jpg aerials. ←

boston | psql -U postgres -d gisdb -h localhost -p 5432

--get a list of raster types supported:

raster2pgsql -G

参数-G输出如下列表：

Available GDAL raster formats:

Virtual Raster

GeoTIFF

National Imagery Transmission Format

Raster Product Format TOC format

ECRG TOC format

Erdas Imagine Images (.img)

CEOS SAR Image

CEOS Image

JAXA PALSAR Product Reader (Level 1.1/1.5)

Ground-based SAR Applications Testbed File Format (.gff)

ELAS

Arc/Info Binary Grid

Arc/Info ASCII Grid

GRASS ASCII Grid

SDTS Raster

DTED Elevation Raster

Portable Network Graphics

JPEG JFIF

In Memory Raster

Japanese DEM (.mem)

Graphics Interchange Format (.gif)

Graphics Interchange Format (.gif)

Envisat Image Format

Maptech BSB Nautical Charts

X11 PixMap Format

MS Windows Device Independent Bitmap

SPOT DIMAP

AirSAR Polarimetric Image

RadarSat 2 XML Product

PCIDSK Database File

PCRaster Raster File

ILWIS Raster Map

SGI Image File Format 1.0

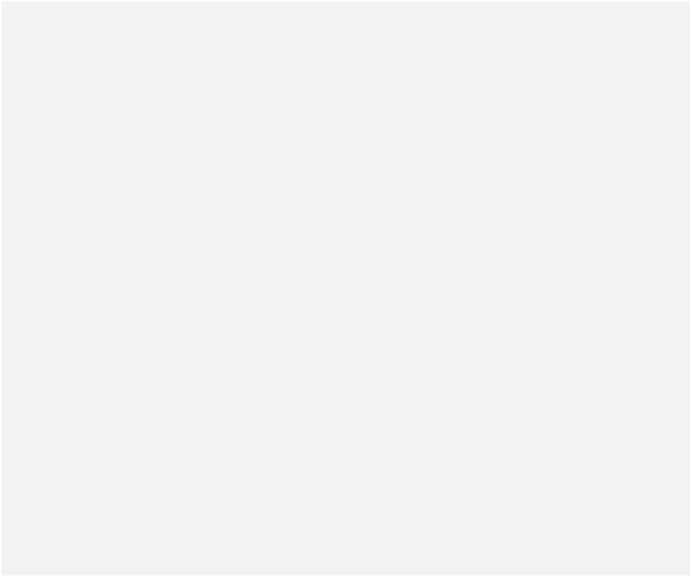
SRTMHGT File Format

Leveller heightfield

Terragen heightfield

USGS Astrogeology ISIS cube (Version 3)

USGS Astrogeology ISIS cube (Version 2)



PostGIS 2.2.0dev 开发手册

61 / 686

NASA Planetary Data System

EarthWatch .TIL

ERMapper .ers Labelled

NOAA Polar Orbiter Level 1b Data Set

FIT Image

GRIdded Binary (.grb)

Raster Matrix Format

EUMETSAT Archive native (.nat)

Idrisi Raster A.1

Intergraph Raster

Golden Software ASCII Grid (.grd)

Golden Software Binary Grid (.grd)

Golden Software 7 Binary Grid (.grd)

COSAR Annotated Binary Matrix (TerraSAR-X)

TerraSAR-X Product

DRDC COASP SAR Processor Raster

R Object Data Store

Portable Pixmap Format (netpbm)

USGS DOQ (Old Style)

USGS DOQ (New Style)

ENVI .hdr Labelled

ESRI .hdr Labelled

Generic Binary (.hdr Labelled)

PCI .aux Labelled

Vexcel MFF Raster

Vexcel MFF2 (HKV) Raster

Fuji BAS Scanner Image

GSC Geogrid

EOSAT FAST Format

VTP .bt (Binary Terrain) 1.3 Format

Erdas .LAN/.GIS

Convair PolGASP

Image Data and Analysis

NLAPS Data Format

Erdas Imagine Raw

DIPEx

FARSITE v.4 Landscape File (.lcp)

NOAA Vertical Datum .GTX

NADCON .los/.las Datum Grid Shift

NTv2 Datum Grid Shift

ACE2

Snow Data Assimilation System

Swedish Grid RIK (.rik)

USGS Optional ASCII DEM (and CDED)

GeoSoft Grid Exchange Format

Northwood Numeric Grid Format .grd/.tab

Northwood Classified Grid Format .grc/.tab

ARC Digitized Raster Graphics

Standard Raster Product (ASRP/USRP)

Magellan topo (.blx)

SAGA GIS Binary Grid (.sdat)

Kml Super Overlay

ASCII Gridded XYZ

HF2/HFZ heightfield raster

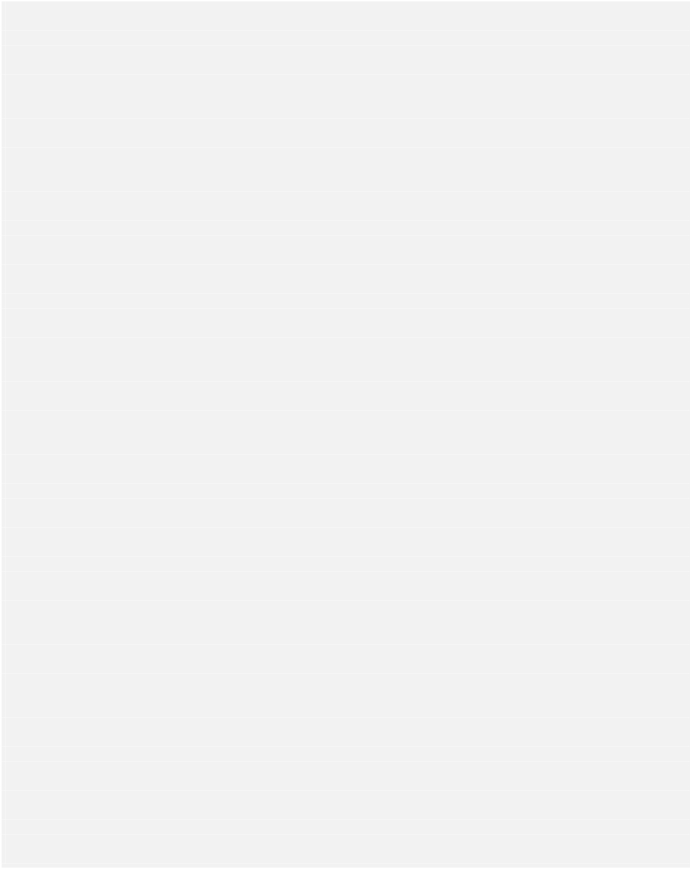
OziExplorer Image File

USGS LULC Composite Theme Grid

Arc/Info Export E00 GRID

ZMap Plus Grid

NOAA NGS Geoid Height Grids



PostGIS 2.2.0dev 开发手册

62 / 686

**5.1.2**

**使用PostGIS栅格函数来创建栅格**

在许多场景下，你会想在数据库中创建栅格即栅格表。PostGIS提供了相当多的函数来干这个事。一般步骤如下：

1. 创建一个带有栅格记录的栅格列的表可以用下面的完成：

CREATE TABLE myrasters(rid serial primary key, rast raster);

2. 有许多函数可以完成这一步骤，如果你创建的栅格不依赖于其他栅格，那么你可以使用函数：ST\_MakeEmptyRaster，接着使用ST\_AddBand

你也可以使用geometry对象来创建栅格。要想实现这个目的，你需要使用函数ST\_AsRaster。可能还需要和其他函数比如函数ST\_Union 或函数ST\_MapAlgebraFct 或者其他地图代数系列函数联合使用。

甚至还有一些根据一些已经存在的栅格表创建新的栅格表的可选函数。例如你可以使用函数ST\_Transform 根据一个已有的栅格表在其他投影系中创建一个新的栅格表。

3.一旦你完成了初始化你的栅格表，你需要在你的栅格列上创建一个空间索引，使用如下语法：

CREATE INDEX myrasters\_rast\_st\_convexhull\_idx ON myrasters USING gist( ST\_ConvexHull( rast) );

注意这里函数ST\_ConvexHull 的使用，因为大多数栅格操作符都是基于栅格的凸包。

**注意**

在PostGIS 2.0版本之前，栅格都是基于envelop（包络线）的而不是凸包。为了让空间索引能够正常使用，你需要丢掉之前的索引，然后使用基于凸包的索引。

4. 使用函数AddRasterConstraints添加约束

**5.2**

**栅格系统目录**

和PostGIS打包在一起的有两个栅格视图。这两个视图都利用嵌入在栅格约束中的信息进行展示。因此这两个视图总是与表中的栅格数据保持一致性兼容，因为表中强加了约束。

1. raster\_columns 这个视图记录了包含你数据库中所有的栅格表的列。

2. raster\_overviews 这个视图记录了包含你的数据库中所有的栅格表的列，并以细粒度表的方式概述。这种类型的表在你使用栅格加载程序并加上-l参数时候将会生成。

**5.2.1**

**栅格列的Catalog（目录）**

raster\_columns是一个包含你的数据库所有的栅格表的栅格类型的列的catalog目录。这个视图利用嵌入在栅格约束中的信息进行展示。因此这两个视图总是与表中的栅格数据保持一致性兼容，即便你从一个其他数据库的备份中恢复栅格表。如果你在创建表时候没有使用加载工具或者使用加载工具时候没有指定-C 标识，你可以使用函数AddRasterConstraints 来强加约束，这样的话raster\_columns目录

就可以把你的栅格瓦片信息进行注册。

· r\_table\_catalog 表示数据表所在的库。它总会读取当前数据库信息

· r\_table\_schema 表示栅格表所属的schema。



PostGIS 2.2.0dev 开发手册

63 / 686

· r\_table\_name 栅格表

· r\_raster\_column 表名为r\_table\_name的栅格类型列。PostGIS中没有设置让你每个表有多个栅格列的障碍，因此有可能一个栅格表会有多个不同的栅格列。

· srid ：栅格表的SRID，可以在4.3.1节了解相关

· scale\_x ：空间参考坐标和像素的缩放比例。只有在栅格列的所有瓦片的scale\_x是相同的才可用，然后会加上这个约束。请参考函数ST\_ScaleX 获取更多细节。

· scale\_y空间参考坐标和像素的缩放比例。只有在栅格列的所有瓦片的scale\_y是相同的才可用，然后会加上这个约束。请参考函数ST\_ScaleY 获取更多细节。

· blocksize\_x 每个栅格瓦片的宽度(像素横向的个数)，参考函数ST\_Width 获取更多细节。

· blocksize\_y 每个栅格瓦片的高度(向下的像素数)。参考 ST\_Height 获取更多细节。

· same\_alignment 如果所有的栅格瓦片有相同的alignment对齐方式，那么返回true。参考函数ST\_SameAlignment 获得更多细节。

· regular\_blocking 如果栅格列有空间唯一和覆盖瓦片的约束，那么参数值应该是TRUE，否则是FALSE.

· num\_bands 你的栅格集合的每个瓦片的波段数。这个与函数ST\_NumBands 返回结果一样。

· pixel\_types 定义每一个波段的pixel类型的数组。这个数组元素的个数和你波段的总数是一样的。这个参数是函数ST\_BandPixelType的参数之一。

· nodata\_values 表示每一个波段的nodata\_value的double型数组。这个数组元素的个数和你波段的总数是一样的。大多数操作中定义每个波段pixel值可以忽略。这个参数和函数ST\_BandNoDataValue返回的信息类似。

· extent ：这个是你的栅格集合的所有栅格的边界。如果你准备加载更多数据，但却会改变栅格的边界，那么你需要运行函数DropRasterConstraints删除约束再加载数据，加载完成后，再运行函数AddRasterConstraints添加约束。

· spatial\_index 如果一个栅格列有空间索引，那么返回值为true。

**5.2.2**

**栅格概视图**

raster\_overviews记录了用于概视图的栅格表的列的信息以及额外的何时需要利用概视图的有益信息。概视图表记录在raster\_columns 和 raster\_overviews中 ，因为这是本身他们的功能，但同样可以用于把低分辨率作为高分辨率的简化的目的。当你在加载数据，使用-l开关时候，他们会沿着主栅格表生成的。

概视图表包含了和其他栅格表一样的约束以及额外的对于概视图的信息。

**注意**

raster\_overviews 里面的信息不是重复raster\_columns里面的信息。如果你想得到raster\_columns里面的的概视图的信息，你可以把raster\_overviews 和raster\_columns 做关联就可以获得全部新。

使用概视图的两个主要理由是：

1.中心表的分辨率的描述通常用于快速的地图放大。



PostGIS 2.2.0dev 开发手册

64 / 686

2. 通常这种情况下会计算比更高分辨率的父瓦片计算更快一些，因为分辨率更低的会有更少的记录数，每个像素会覆盖更多的区域。虽然计算没有更高分辨率的精确，但是在许多经验规则下通常这种分辨率是足够的了。

raster\_overviews 包含了如下列的信息

· o\_table\_catalog 概视图表所在的数据库。它总是读取当前数据库信息。

· o\_table\_schema ：栅格表概视图所属的数据库schema

· o\_table\_name 栅格概视图表名

· o\_raster\_column 概视图表的栅格列

· r\_table\_catalog 概视图服务的栅格表所在的数据库。它总是读取当前数据库信息。

· r\_table\_schema 概视图提供服务的栅格表所在的数据库schema。

· r\_table\_name 概视图服务的栅格表名

· r\_raster\_column 概视图提供信息的栅格列

· overview\_factor – 这个是概视图表的金字塔级别。这个值越大，概视图表的分辨率越低。如果给定了一个图片的目录，加载工具raster2pgsql 会计算每个图片文件的概视图，然后单独加载。这个参数默认值是Level 1，并且针对的是原始的文件。Level 2 会让每个瓦片代表4个原始文件。因此举个例子，如果你有一个包含5000x5000 像素图片的文件夹，并且打算分成每个125x125像素的块，因此对于你的基础表的每个图像文件，你将会得到 (5000\*5000)/(125\*125) = 1600条记录，你的 (level=2) o\_2 表将会有ceiling(1600/Power(2,2)) = 400 条记录，你的表(level=3) o\_3 将会有ceiling(1600/Power(2,3) ) = 200 条记录。如果你的像素size不能用你的瓦片大小除尽，你会得到一些碎片瓦片（瓦片没有完全填满）。注意raster2pgsql生成的每个概视图瓦片都和瓦片的父级瓦片有相同的像素个数，但是分辨率更低，因为这个瓦片的每个像素相当于代表原始的Power(2,overview\_factor) 这么多个像素。

**5.3**

**使用PostGIS 栅格创建定制应用**

PostGIS栅格为你提供了许多SQL函数用于把栅格与许多已知格式的数据进行交换，事实上这让你有了许多数据交换的选择。例如你可以使用办公工具OpenOfﬁce / LibreOfﬁce 来展示栅格数据，参考 [Rendering PostGIS](http://www.postgresonline.com/journal/archives/244-Rendering-PostGIS-Raster-graphics-with-LibreOffice-Base-Reports.html) [Raster graphics with LibreOfﬁce Base Reports.](http://www.postgresonline.com/journal/archives/244-Rendering-PostGIS-Raster-graphics-with-LibreOffice-Base-Reports.html)除此之外，你可以使用一些广泛使用的语言来演示本节的样例。

**5.3.1**

**在PHP中使用函数函数ST\_AsPNG和其他栅格函数输出栅格的样例**

本节，我们将会展示怎样使用PHP PostgreSQL驱动和ST\_AsGDALRaster 系列函数来输出一个栅格的波段1,2,3到一个PHP请求数据流中，这样就可以把它嵌入到一个HTML标签img 的src属性中。

在这个样例中，你需要使用驱动npgsql .NET PostgreSQL，你可以从这里获取到最新的驱动版本http://npgsql.projects.postgresql.o。你只需要下载最新的版本并且把它丢进ASP.NET的bin目录就可以了。下面的查询样例展示了怎样联合使用栅格的一系列函数来获取到一个特殊的wgs 84缓冲区相交的所有瓦片，然后使用函数ST\_Union对相交的瓦片求并集，然后把返回的所有波段使用函数ST\_Transform转换到用户自定义的投影系中，最后再使用函数ST\_AsPNG 把它输出到一个PNG图片文件中。

你会调用下面的URL

http://mywebserver/test\_raster.php?srid=2249

来获取Massachusetts state plane feet（Massachusetts州标准单位）的栅格图像

<?php

/\*\* contents of test\_raster.php \*\*/

$conn\_str =’dbname=mydb host=localhost port=5432 user=myuser password=mypwd’;

$dbconn = pg\_connect($conn\_str);

header(’Content-Type: image/png’);

PostGIS 2.2.0dev 开发手册

65 / 686

/\*\*If a particular projection was requested use it otherwise use mass state plane meters

\*\*/

if (!empty( $\_REQUEST[’srid’] ) && is\_numeric( $\_REQUEST[’srid’]) ){

$input\_srid = intval($\_REQUEST[’srid’]);

}

else { $input\_srid = 26986; }

/\*\* The set bytea\_output may be needed for PostgreSQL 9.0+, but not for 8.4 \*\*/

$sql = "set bytea\_output=’escape’;

SELECT ST\_AsPNG(ST\_Transform(

ST\_AddBand(ST\_Union(rast,1), ARRAY[ST\_Union(rast,2),ST\_Union(rast,3)])

,$input\_srid) ) As new\_rast

FROM aerials.boston

WHERE

←

ST\_Intersects(rast, ST\_Transform(ST\_MakeEnvelope(-71.1217, 42.227, -71.1210,

42.218,4326),26986) )";

$result = pg\_query($sql);

$row = pg\_fetch\_row($result);

pg\_free\_result($result);

if ($row === false) return;

echo pg\_unescape\_bytea($row[0]);

?>

←

**5.3.2**

**在ASP.NET C#中使用函数函数ST\_AsPNG和其他栅格函数输出栅格的样例**

本节，我们将会展示怎样使用Npgsql PostgreSQL .NET 驱动和ST\_AsGDALRaster 系列函数来输出一个栅格的波段1,2,3到一个PHP请求数据流中，这样就可以把它嵌入到一个HTML标签img 的src属性中。

在这个样例中，你需要使用驱动npgsql .NET PostgreSQL，你可以从这里获取到最新的驱动版本http://npgsql.projects.postgresql.o。你只需要下载最新的版本并且把它丢进ASP.NET的bin目录就可以了。下面的查询样例展示了怎样联合使用栅格的一系列函数来获取到一个特殊的wgs 84缓冲区相交的所有瓦片，然后使用函数ST\_Union对相交的瓦片求并集，然后把返回的所有波段使用函数ST\_Transform转换到用户自定义的投影系中，最后再使用函数ST\_AsPNG 把它输出到一个PNG图片文件中。

下面的样例和5.3.1节类似，只是使用了C#进行实现：

你会调用下面的URL

http://mywebserver/TestRaster.ashx?srid=2249

来获取Massachusetts state plane feet（Massachusetts州标准单位）的栅格图像

-- web.config connection string section --

<connectionStrings>

<add name="DSN"

connectionString="server=localhost;database=mydb;Port=5432;User Id=myuser;password= ←

mypwd"/>

</connectionStrings>

// Code for TestRaster.ashx

<%@ WebHandler Language="C#" Class="TestRaster" %>

using System;

using System.Data;

using System.Web;

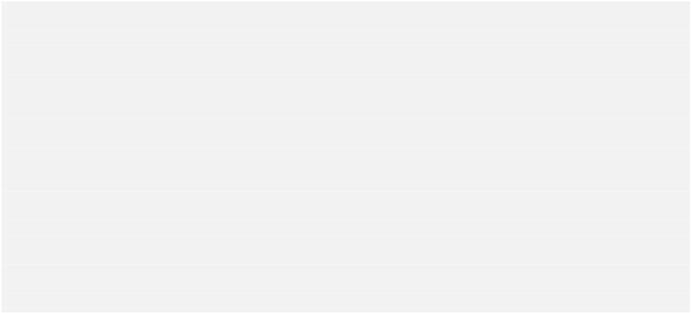
using Npgsql;

public class TestRaster : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{



PostGIS 2.2.0dev 开发手册

66 / 686

context.Response.ContentType = "image/png";

context.Response.BinaryWrite(GetResults(context));

}

public bool IsReusable {

get { return false; }

}

public byte[] GetResults(HttpContext context)

{

byte[] result = null;

NpgsqlCommand command;

string sql = null;

int input\_srid = 26986;

try {

using (NpgsqlConnection conn = new NpgsqlConnection(System.Configuration. ←

ConfigurationManager.ConnectionStrings["DSN"].ConnectionString)) {

conn.Open();

if (context.Request["srid"] != null)

{

input\_srid = Convert.ToInt32(context.Request["srid"]);

}

sql = @"SELECT ST\_AsPNG(

ST\_Transform(

ST\_AddBand(

ST\_Union(rast,1), ARRAY[ST\_Union(rast,2),ST\_Union(rast,3)])

,:input\_srid) ) As new\_rast

FROM aerials.boston

WHERE

ST\_Intersects(rast,

ST\_Transform(ST\_MakeEnvelope(-71.1217, 42.227,

←

-71.1210, 42.218,4326),26986) )";

command = new NpgsqlCommand(sql, conn);

command.Parameters.Add(new NpgsqlParameter("input\_srid", input\_srid));

result = (byte[]) command.ExecuteScalar();

conn.Close();

}

}

catch (Exception ex)

{

result = null;

context.Response.Write(ex.Message.Trim());

}

return result;

}

}

**5.3.3**

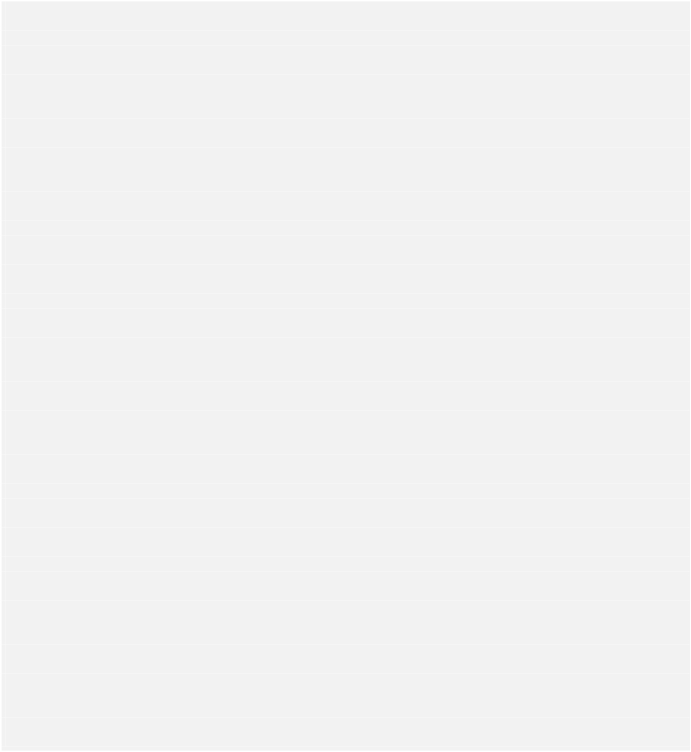
**将栅格查询通过Java控制台程序输出成图像文件**

这是一个简单的Java控制台程序，先做返回一个图像的查询，然后输出成指定文件。

你可以在如下链接下载最新的PostgreSQL JDBC 驱动<http://jdbc.postgresql.org/download.html>

你可以通过如下命令“编译”自己的Java代码：

set env CLASSPATH .:..\postgresql-9.0-801.jdbc4.jar



PostGIS 2.2.0dev 开发手册

67 / 686

javac SaveQueryImage.java

jar cfm SaveQueryImage.jar Manifest.txt \*.class

使用命令调用这个函数方法如下：

java -jar SaveQueryImage.jar "SELECT ST\_AsPNG(ST\_AsRaster(ST\_Buffer(ST\_Point(1,5),10, ’ ←

quad\_segs=2’),150, 150, ’8BUI’,100));" "test.png"

-- Manifest.txt --

Class-Path: postgresql-9.0-801.jdbc4.jar

Main-Class: SaveQueryImage

// Code for SaveQueryImage.java

import java.sql.Connection;

import java.sql.SQLException;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.io.\*;

public class SaveQueryImage {

public static void main(String[] argv) {

System.out.println("Checking if Driver is registered with DriverManager.");

try {

//java.sql.DriverManager.registerDriver (new org.postgresql.Driver());

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

}

catch (ClassNotFoundException cnfe) {

System.out.println("Couldn’t find the driver!");

cnfe.printStackTrace();

System.exit(1);

}

Connection conn = null;

try {

conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydb","myuser ←

", "mypwd");

conn.setAutoCommit(false);

PreparedStatement sGetImg = conn.prepareStatement(argv[0]);

ResultSet rs = sGetImg.executeQuery();

FileOutputStream fout;

try

{

rs.next();

/\*\* Output to file name requested by user \*\*/

fout = new FileOutputStream(new File(argv[1]) );

fout.write(rs.getBytes(1));

fout.close();

}

catch(Exception e)

{

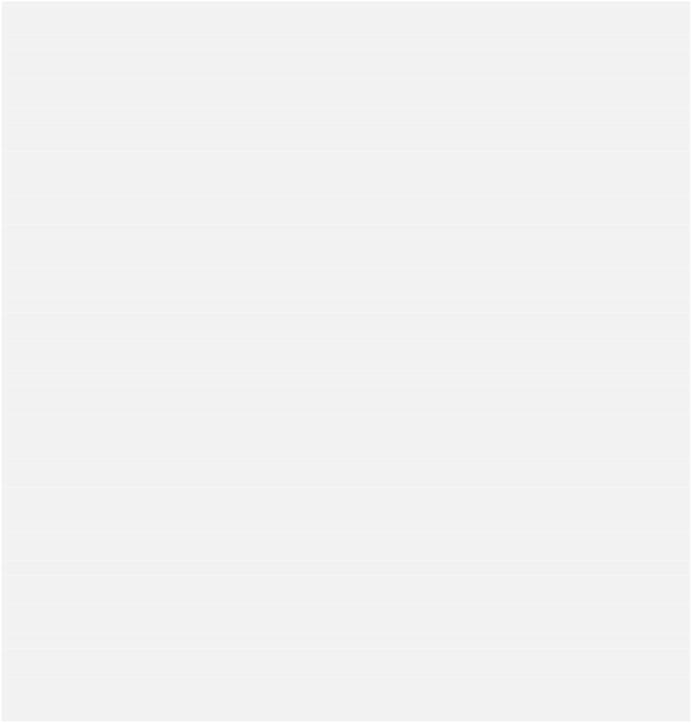
System.out.println("Can’t create file");

e.printStackTrace();

}

rs.close();

sGetImg.close();



PostGIS 2.2.0dev 开发手册

68 / 686

conn.close();

}

catch (SQLException se) {

System.out.println("Couldn’t connect: print out a stack trace and exit.");

se.printStackTrace();

System.exit(1);

}

}

}

**5.3.4**

**通过SQL使用PLPython 来导出image图像**

这是一个plpython语言函数，它对每一行记录在服务器目录上创建一个文件。条件是你需要一件安装了plpython扩展的支持。

这个函数可以在plpythonu和plpython3u(Python版本是3.x)下使用。

CREATE OR REPLACE FUNCTION write\_file (param\_bytes bytea, param\_filepath text)

RETURNS text

AS $$

f = open(param\_filepath, ’wb+’)

f.write(param\_bytes)

return param\_filepath

$$ LANGUAGE plpythonu;

--write out 5 images to the PostgreSQL server in varying sizes

-- note the postgresql daemon account needs to have write access to folder

-- this echos back the file names created;

SELECT write\_file(ST\_AsPNG(

ST\_AsRaster(ST\_Buffer(ST\_Point(1,5),j\*5, ’quad\_segs=2’),150\*j, 150\*j, ’8BUI’,100)),

’C:/temp/slices’|| j || ’.png’)

FROM generate\_series(1,5) As j;

write\_file

---------------------

C:/temp/slices1.png

C:/temp/slices2.png

C:/temp/slices3.png

C:/temp/slices4.png

C:/temp/slices5.png

**5.3.5**

**使用PSQL输出栅格**

不幸的是PSQL没有简单的内置功能来输出二进制栅格数据。虽然有点hack，但基于如下文章的建议Clever Trick Challenge -- Outputting bytea with psql (<http://people.planetpostgresql.org/andrew/index.php?/archives/196-Clever-trick-challenge.html>,笔者注：这个已经打不开了，原文提供的如此)，可以提供对大型对象导出的支持。要使用这个方法，首先通过命令行方式连接数据库。不像Python方法，这个方法在你本地的电脑上创建文件。

SELECT oid, lowrite(lo\_open(oid, 131072), png) As num\_bytes

FROM

( VALUES (lo\_create(0),

ST\_AsPNG( (SELECT rast FROM aerials.boston WHERE rid=1) )

) ) As v(oid,png);

-- you’ll get an output something like --

oid

| num\_bytes

---------+-----------

2630819 | 74860

-- next note the oid and do this replacing the c:/test.png to file path location

PostGIS 2.2.0dev 开发手册

69 / 686

-- on your local computer

\lo\_export 2630819 ’C:/temp/aerial\_samp.png’

-- this deletes the file from large object storage on db

SELECT lo\_unlink(2630819);