5 / 686

# Chapter 2

**PostGIS 安装**

本章详细介绍安装PostGIS必须的步骤.

**2.1**

**安装概要**

假设当前使用路径中有所有的依赖关系，编译命令如下

tar xvfz postgis-2.2.0dev.tar.gz

cd postgis-2.2.0dev

./configure

make

make install

安装PostGIS完成后，你需要在每个需要使用该功能的数据库中安装该插件

**注意：**

栅格支持是当前可选的，但默认安装。但对于使用PostgreSQL 9.1 及之后的版本，使用插件模式安装时候，模型栅格是默认安装的。插件模式安装方法是首选的，更方便用户。为了创建有空处理功能的数据库，执行下面的命令：

psql -d yourdatabase -c "CREATE EXTENSION postgis;"

psql -d yourdatabase -c "CREATE EXTENSION postgis\_topology;"

psql -d yourdatabase -c "CREATE EXTENSION postgis\_tiger\_geocoder;"

请参阅第2.4.3详情关于如何进行PostGIS插件的安装及启用，升级或者不使用非插件安装的方式而使用插件安装的方式

对于那些运行PostgreSQL 9.0版本数据库或者因为某些原因不愿意使用栅格功能支持或者仅仅是因为过时想法的原因，

这里有更长更痛苦的安装指南：

所有的sql 文件一旦执行，将会同时安装（拷贝）在将安装你的PostgreSQL安装目录下的share/contrib/postgis-2.2文件夹下面

例如译者的如下：

createdb yourdatabase

createlang plpgsql yourdatabase

psql -d yourdatabase -f postgis.sql

psql -d yourdatabase -f postgis\_comments.sql

psql -d yourdatabase -f spatial\_ref\_sys.sql

psql -d yourdatabase -f rtpostgis.sql

psql -d yourdatabase -f raster\_comments.sql

psql -d yourdatabase -f topology/topology.sql

psql -d yourdatabase -f topology\_comments.sql

The rest of this chapter goes into detail each of the above installation steps

.

本章如下的部分将详细介绍上述被一个安装步骤



6 / 686

**2.2**

**安装要求**

PostGIS 在安装和使用前有如下要求:

要求

· PostgreSQL 9.1 或更高版本.必须完整的安装PostgreSQL包括服务器端，PostgreSQL可以到如下网站获取

http://www.postgresql.org .

PostgreSQL与PostGIS 版本支持(兼容)关系和PostGIS与GEOS 版本支持(兼容)关系请参考 [http://trac.osgeo.org/postgis/wiki/-UsersWikiPostgreSQLPostGIS](http://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQLPostGIS)

· GNU C 编译器 (gcc编译器). 一些其他的ANSI C 编译器也能用来编译PostGIS，但是我们发现用gcc编译器出现的问题要少得多。

· GNU Make (gmake or make).对于许多系统，GNU 版本的make是默认的。通过如下命令检查版本：make –v，其他版本可能会不正确处理PostGIS Makefile文件。

· Proj4 重投影的库4.6.0或更高. Proj4 库用于为PostGIS提供坐标投影系统转换的库支持. Proj4可从如下网站下载http://trac.osgeo.org/proj/ .

· GEOS几何对象库,版本3.3或更高，但推荐3.4 及以上版本，这样可以充分利用所有的新功能及特性，不使用GEOS 3.4及以上版本，你会错过一些重大的改进，如ST\_Triangles和长时间运行事务中断功能，以及改善几何类型对象的校验和修复几何类型对象的函数，如st\_validdetail和st\_makevalid。GEOS 3.3.2 +也是拓扑功能支持所必须的版本。GEOS在如下网站可供下载[http://trac.osgeo.org/geos / 3.4](http://trac.osgeo.org/geos%20/%203.4)级以上版本是对老版本相当安全的兼容升级

· LibXML2, 版本 2.5.x 或更高. LibXML2在当前某些输入性函数例如 (ST\_GeomFromGML and ST\_GeomFromKML)中使用。

LibXML2 在如下网站可以提供下载 [http://xmlsoft.org/downloads.html.](http://xmlsoft.org/downloads.html)

· JSON-C, 版本 0.9 或更高. JSON-C 当前在通过函数ST\_GeomFromGeoJson 导入GeoJSON数据需要使用，可以在如下网站获取

[https://github.com/json-c/json-c/releases/.](https://github.com/json-c/json-c/releases)

· GDAL, 版本 1.8 或更高 (1.9 或更高强烈推荐，因为在更低版本一些功能不能很好的运行或者运行结果有不同。栅格功能支持必须需要这个库，对那些运行9.1版本及以上的PostgreSQL数据库，同时想使用CREATE EXTENSION postgis这种方式安装的高度推荐该库。下载地址：

[http://trac.osgeo.org/gdal/wiki/DownloadSource.](http://trac.osgeo.org/gdal/wiki/DownloadSource)

可选包

· GDAL (伪可选(译者注：意思是如果要使用某些功能，这个还是必选的))：只有你不想要栅格功能，并且不在意使用CREATE EXTENSION postgis 方式安装postgis时候，你可以省略这个库。记住其他插件可能需要postgis扩展，除非你安装postgis扩展，否则可能安装这些插件不会成功。因此强烈建议安装时候添加 GDAL 支持

· GTK (需要 GTK+2.0, 2.8+及以上)为了编译 shp2pgsql-gui shape 文件加载工具. http://www.gtk.org/ .

· SFCGAL, 版本 1.0 (或更高)

可以为PostGIS提供额外的2D和3D的高级分析功能支持。请参考8.10节。也允许使用SFCGAL而不是GEOS提供的一些后端2D函数（例如

ST\_Intersection或ST\_Area）。如果安装了SFCGAL，PostgreSQL配置变量postgis.backend允许用户(相对于后端或者说服务端)来控制他想使用哪个服务端库，默认使用GEOS库。注意:SFCGAL 1.0需要至少CGAL 4.1库以及Boost 库1.46版本的支持

(参考: http://oslandia.github.io/SFCGAL/installation.html) [https://github.com/Oslandia/SFCGAL.](https://github.com/Oslandia/SFCGAL)

· CUnit (CUnit). 回归测试需要这个库.<http://cunit.sourceforge.net/>

· Apache Ant (ant) Java目录下的任何驱动打包、编译都需要这个工具，可以去如下网站获取http://ant.apache.org

· DocBook (xsltproc)：创建文档需要这个包。可以去如下网站获取http://www.docbook.org/ .

· DBLatex (dblatex) 创建PDF格式的文档需要这个包. 可以去如下网站获取 http://dblatex.sourceforge.n.

· ImageMagick (convert)：生成文档中的图片需要这个包. 可以去如下网站获取

<http://www.imagemagick.org/>

7 / 686

**2.3**

**获取源码**

从如下网站获取PostGIS源码包<http://postgis.net/stuff/postgis-2.2.0dev.tar.gz>

wget http://postgis.net/stuff/postgis-2.2.0dev.tar.gz

tar -xvzf postgis-2.2.0dev.tar.gz

上述命令会在当前目录下生成一个名为postgis-2.2.0dev的子目录

另一种方式是使用svn 从svn仓库中http://svn.osgeo.org/postgis/trunk/检出(checkout)出源码.

svn checkout http://svn.osgeo.org/postgis/trunk/ postgis-2.2.0dev

切换至刚才解压源码包后新创建的postgis-2.2.0dev目录，继续安装

**2.4**

**用源码编译并安装:详细步骤**

**注意：**

许多操作系统现在已经预安装了PostgreSQL/PostGIS包。在许多情况下，如果你想使用最前沿的版本，你才编译或者你是一个软件包的维护者。本节包含一般编译说明，如果您正在编译Windows或其他操作系统，您可以在[PostGIS User contributed compile guides](http://trac.osgeo.org/postgis/wiki/UsersWikiInstall)和 [PostGIS Dev Wiki.](http://trac.osgeo.org/postgis/wiki/DevWikiMain)找到额外的更详细的帮助。不同操作系统预安装的包在如下地址列出： [PostGIS Pre-built Packages](http://trac.osgeo.org/postgis/wiki/UsersWikiPackages)。如果你是一个Windows用户，你可以通过Stackbuilder或[PostGIS Windows 下载地址](http://www.postgis.org/download/windows/experimental.php)。我们还有一个[最新的Windows试验版本](http://postgis.net/windows_downloads/)，这些版本通常一周更新一次或两次或任何时候进行一些令人激动的功能更新。你可以根据PostGIS进展发布使用这些功能。

PostGIS模块是PostgreSQL服务器端的一个扩展。因此，为了编译PostGIS 2.2.0dev，必须拥有完整的PostgreSQL

服务器头文件访问权限。这个版本的PostGIS可以在PostgreSQL版本9.1或更高的版本编译。不支持早期版本PostgreSQL数据库

如果你还没有安装PostgreSQL，参阅PostgreSQL安装指南。http://www.postgresql.org .

**注意：**

关于 GEOS 功能,当年安装PostgresSQL时候，你可能需要显式地使用C++标准的库来链接（译者注：C/C++代码需要经过编译和链接两个阶段才能生成可执行程序）:

LDFLAGS=-lstdc++ ./configure [YOUR OPTIONS HERE]

这是一个处理使用更老的开发工具产生的一个“虚假”的C++异常解决方案。如果你遇到过这些奇怪的

问题（后端意外关闭或类似的东西）尝试这个小技巧。这将当然需要重新编译你的PostgreSQL

从零开始

如下步骤概述PostGIS源码安装时的配置和编译参数。这些都是针对Linux用户的，不适用Windows或Mac用户。

**2.4.1**

**配置参数**

正如大多数Linux程序安装一样，第一步需要生成文件Makefile，这个文件用于编译源码。使用下面的shell脚本来完成这个事情。

./conﬁgure





8 / 686

没有更多的参数，这个命令将会在你的系统上尝试自动寻找所需要的组件和所需的库来编译PostGIS源码。

虽然这是./ conﬁgure的最常见的用法。这个脚本也接受那些编译PostGIS源码所需要的包和库不在标准位置的其他参数。

下面的列表只显示最常用的参数。想得到一个完整的清单，使用参数—help或--help=short parameters 两种方式获取完整的参数列表

译者注：即为：./configure --help=short parameters 或者./configure –help)

--preﬁx=PREFIX 这是PostGIS库和SQL脚本要安装的目录位置，默认该位置是和PostgreSQL安装位置一样

**警告**

目前该参数不能使用，因为安装包只会把程序安装在PostgreSQL的安装目录中，访问

<http://trac.osgeo.org/postgis/ticket/635>跟踪这个bug

--with-pgconﬁg=FILE PostgreSQL 提供了一个实用的命令叫做pg\_conﬁg 来让一些扩展比如PostGIS来定位PostgreSQL的安装目录。使用这个参数（--with-pgconﬁg=/path/to/pg\_conﬁg）来手动指定PostGIS安装时依赖的PostgreSQL的安装位置

--with-gdalconﬁg=FILE GDAL一个必须的库，提供栅格功能支持 ，gdal-config命令用来定位GDAL库的安装目录，使用该参数如 (--with-gdalconﬁg=/path/to/gdal-conﬁg) 来手动指定PostGIS安装时依赖的GDAL的安装位置

--with-geosconﬁg=FILE GEOS一个必须的库，提供几何类型库支持，, geos-conﬁg命令用来定位GEOS库的安装目录，使用该参数如 (--with-geosconﬁg=/path/to/geos-conﬁg) 来手动指定PostGIS安装时依赖的GEOS的安装位置

--with-xml2conﬁg=FILE LibXML 是一个处理GeomFromKML/GML 所必须的库。如果你安装了libxml，那么就可以找到，如果没有或者你需要一个你特殊使用版本，你需要让PostGIS指向xml2-config命令来定位LibXML的安装目录。使用该参数如 (--with-xml2conﬁg=/path/to/xml2-conﬁg) 来手动指定PostGIS安装时依赖的LibXML的安装位置

--with-projdir=DIR Proj4 是PostGIS所需要的一个投影库。使用该参数如 (--with-projdir=/path/to/projdir) 来手动指定PostGIS安装时依赖的Proj4的安装位置

--with-libiconv=DIR iconv程序的安装目录

--with-jsondir=DIR JSON-C是MIT授权的JSON 库，是PostGIS函数ST\_GeomFromJSON 所需要的。使用该参数如 (--with-jsondir=/path/to/jsondir) 来手动指定PostGIS安装时依赖的JSON-C的安装位置

--with-gui 用于添加GUI对导入数据的支持 (需要 GTK+2.0)。该参数会为命令shp2pgsql创建一个shp2pgsql-gui 图形化页面

--with-raster 添加栅格功能支持。这会构建rtpostgis-2.2.0dev 库和rtpostgis.sql 文件。在PostGIS最终版本（译者注：当前是开发版）中可能不需要该参数，因为默认会添加栅格功能支持。

--with-topology 添加拓扑功能支持。该参数会生成topology.sql 文件。在postgis-2.2.0dev库中没有拓扑功能所需要的所有对应的库

--with-gettext=no 默认PostGIS会试着检测gettext支持并添加编译支持。然而如果你编译时候出现了编译问题导致终止，你可以不加上该参数。参考<http://trac.osgeo.org/postgis/ticket/748> 获取解决该问题的样例。注意：关掉这个参数，你不会丢失很多功能。这个只是用于GUI 装载功能的帮助功能，目前没有相关的帮助文档并且也只是试验功能



9 / 686

**注意**

如果你通过SVN repository 获取PostGIS，第一步真正要做的是执行命令

**./autogen.sh**

该脚本会生成configure命令脚本，并用于定制PostGIS的安装。

如果你通过tar包来获取PostGIS，没必要执行脚本**./autogen.sh**，因为configure脚本已经生成了

**2.4.2**

**构建程序包**

一旦Makefile已经生成了，构建PostGIS是很容易的事情，只需要运行

make

上面的命令最后输出行应该是"PostGIS was built successfully.Read to

install."

从PostGIS v1.4.0版本起,所有的函数都会从文档中生成注释。如果你想安装这些注释到你的空间数据库中，执行需要docbook库的命令：make comments

postgis\_comments.sql 和 其他包注释的脚本文件raster\_comments.sql, topology\_comments.sql 也打包在tar.gz包的doc目录下面。

make comments

PostGIS 2.0引入的东西：make cheatsheets

该命令会生成HTML文档，用于新手参考学习。该命令需要xsltproc库来生成4个doc目录下的HTML文件

topology\_cheatsheet.html, tiger\_geocoder\_cheatsheet.html, raster\_cheatsheet.html, postgis\_cheatsheet.html

你可以下载一些可用的预发布HTML和PDF文档 [PostGIS / PostgreSQL Study Guides](http://www.postgis.us/study_guides)

执行命令：make cheatsheets生成

**2.4.3**

**构建PostGIS插件扩展并部署**

如果你使用的是PostgreSQL 9.1+版本数据库，PostGIS 扩展是自动构建和安装的

如果你从源码构建的，你需要首先生成函数的描述。如果你安装了docbook库，这些会生成。你也可以手动使用如下语句构建

make comments

如果你的程序是tar包，生成函数的注释不是必须要的，因为这些已经在tar 包中预安装了

如果你打包时候依赖PostgreSQL 9.1，自动安装扩展成为make install进程的一个步骤。

当然如果需要的情况下可以通过扩展目录来构建，或者在一个不同的服务器上面通过拷贝的文件来构

cd extensions

cd postgis

make clean

make

make install

cd ..

cd postgis\_topology

make clean

make

make install

无论操作系统是怎样的，对于同一个版本的PostGIS，扩展文件是相同的，因此从一种操作系统拷贝扩展文件到另一种都是可以的，只要你在你的服务器上面安装了PostGIS的二进制程序。如果你想在除了你的开发机器职位的其他服务器上面手动安装扩展，你需要从扩展程序的目录拷贝如下文件到你的目的机器上的PostgreSQL安装目录下的/ share / extension目录以及所需要的PostGIS二进制程序（如果目的机器没有安装PostGIS）



10 / 686

·如果没有特别指出，下面都是表面扩展的一些信息比如版本之类的控制文件postgis.control, postgis\_topology.control.

· 每个扩展下的目录 /sql 下的文件。注意：这些需要拷贝到PostgreSQLD 目录share/extension

extensions/postgis/sql/\*.sql, extensions/postgis\_topology/sql/\*.sql 这些文件要拷贝到PostgreSQLD 目录share/extension中

一旦你完成了这些，你需要在PgAdmin—>extensions中查看postgis, postgis\_topology 是可用的扩展

如果你使用psql，你可以通过如下查询来确认扩展是否已经安装了

SELECT name, default\_version,installed\_version

FROM pg\_available\_extensions WHERE name LIKE ‘postgis%’ ;

name

| default\_version | installed\_version

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

postgis

| 2.2.0dev

| 2.2.0dev

postgis\_topology | 2.2.0dev

|

如果你的查询结果表示你已经在数据库中安装了扩展，你可以看到在查询出的列installed\_version 中看到.

如果你没有查询到记录，这意味着你没有在你的服务器上面安装好PostGIS扩展。PgAdmin III 1.14+ 也会在数据库浏览树形目录下提示这一信息，甚至可以通过右击来运行升级或卸载扩展

如果你已经让扩展可用，你可以有两种方式来在你的数据库中安装PostGIS扩展：使用PgAdmin扩展接口或者使用SQL命令

CREATE EXTENSION postgis;

CREATE EXTENSION postgis\_topology;

CREATE EXTENSION postgis\_tiger\_geocoder;

在你的psql中，你可以使用你安装了什么版本的PostGIS以及他们在哪个schema中安装：

\connect mygisdb

\x

\dx postgis\*

List of installed extensions

-[ RECORD 1 ]-------------------------------------------------

-

Name

Version

Schema

| postgis

| 2.2.0dev

| public

Description | PostGIS geometry, geography, and raster spat..

-[ RECORD 2 ]-------------------------------------------------

-

Name

Version

Schema

| postgis\_tiger\_geocoder

| 2.2.0dev

| tiger

Description | PostGIS tiger geocoder and reverse geocoder

-[ RECORD 3 ]-------------------------------------------------

-

Name

Version

Schema

| postgis\_topology

| 2.2.0dev

| topology

Description | PostGIS topology spatial types and functions



11 / 686

**警告**

扩展的系统表spatial\_ref\_sys, layer, topology 不能精确地备份。他们只有在各自的postgis或postgis\_topology扩展备份时候才能备份，而这种似乎只有在你备份整个数据库的时候才会发生。从

PostGIS 2.0.1开始，当数据库被备份的时候，只有没有和PostGIS一起的SRID记录才会被备份。所以,不要随便修改我们放在PostGIS spatial\_ref\_sys系统表内的SRID值并期望你能更改成功。如果你发现了问题，在跟踪bug的网站上发一个跟踪单即可。扩展的系统表结构永远不能够备份，因为他们是使用CREATE EXTENSION方式来创建的，并被认定和提供的扩展版本提供的一致。这些都内置在当前PostgreSQL扩展安装方式中，因此我们对它什么都不必做

如果你没有使用超赞的创建扩展的方式安装PostGIS 2.2.0dev版本，你可以通过首先通过运行如下脚本来把你的PostGIS版本升级到最新的小版本，然后可以基于创建扩展的方式来安装PostGIS：postgis\_upgrade\_22\_minor.sql,raster\_upgrade\_22\_minor.sql,topology\_upgrade\_22\_minor.sql.

如果你安装PostGIS时候没有添加栅格支持，你需要首先安装栅格功能支持（通过运行脚本rtpostgis.sql）

然后你需要执行下面的命令来创建各自的扩展

CREATE EXTENSION postgis FROM unpackaged;

CREATE EXTENSION postgis\_topology FROM unpackaged;

CREATE EXTENSION postgis\_tiger\_geocoder FROM unpackaged;

**2.4.4**

**测试**

如果你想测试你打包的PostGIS，执行下面命令

make check

上面的命令会使用生成的库在PostgreSQL数据库上运行各种检查和回归测试

**注意**

如果你使用的是非标准的PostgreSQL, GEOS, 或 Proj4 位置来configure你的PostGIS程序，你需要在环境变量LD\_LIBRARY\_PATH里面来配置他们的库位置

**警告**

目前命令**make check** 依赖环境变量PATH 和 PGPORT。它不需要使用PostgreSQL的具体版本，因此确保修改了你的PATH环境变量是正确的，即该变量可以正确检测到你的PostgreSQL的安装目录，以便在配置或者处理一些蛋疼的问题

如果运行成功，测试输出会和如下的输出类似

CUnit - A Unit testing framework for C - Version 2.1-0

http://cunit.sourceforge.net/

Suite: print\_suite

Test: test\_lwprint\_default\_format ... passed

Test: test\_lwprint\_format\_orders ... passed

Test: test\_lwprint\_optional\_format ... passed

Test: test\_lwprint\_oddball\_formats ... passed

Test: test\_lwprint\_bad\_formats ... passed

Suite: misc

Test: test\_misc\_force\_2d ... passed







12 / 686

Test: test\_misc\_simplify ... passed

Test: test\_misc\_count\_vertices ... passed

Test: test\_misc\_area ... passed

Test: test\_misc\_wkb ... passed

Suite: ptarray

Test: test\_ptarray\_append\_point ... passed

Test: test\_ptarray\_append\_ptarray ... passed

Test: test\_ptarray\_locate\_point ... passed

Test: test\_ptarray\_isccw ... passed

Test: test\_ptarray\_signed\_area ... passed

Test: test\_ptarray\_desegmentize ... passed

Test: test\_ptarray\_insert\_point ... passed

Test: test\_ptarray\_contains\_point ... passed

Test: test\_ptarrayarc\_contains\_point ... passed

Suite: PostGIS Computational Geometry Suite

Test: test\_lw\_segment\_side ... passed

Test: test\_lw\_segment\_intersects ... passed

Test: test\_lwline\_crossing\_short\_lines ... passed

Test: test\_lwline\_crossing\_long\_lines ... passed

Test: test\_lwline\_crossing\_bugs ... passed

Test: test\_lwpoint\_set\_ordinate ... passed

Test: test\_lwpoint\_get\_ordinate ... passed

Test: test\_point\_interpolate ... passed

Test: test\_lwline\_clip ... passed

Test: test\_lwline\_clip\_big ... passed

Test: test\_lwmline\_clip ... passed

Test: test\_geohash\_point ... passed

Test: test\_geohash\_precision ... passed

Test: test\_geohash ... passed

Test: test\_geohash\_point\_as\_int ... passed

Test: test\_isclosed ... passed

Suite: buildarea

Test: buildarea1 ... passed

Test: buildarea2 ... passed

Test: buildarea3 ... passed

Test: buildarea4 ... passed

Test: buildarea4b ... passed

Test: buildarea5 ... passed

Test: buildarea6 ... passed

Test: buildarea7 ... passed

Suite: clean

Test: test\_lwgeom\_make\_valid ... passed

Suite: PostGIS Measures Suite

Test: test\_mindistance2d\_tolerance ... passed

Test: test\_rect\_tree\_contains\_point ... passed

Test: test\_rect\_tree\_intersects\_tree ... passed

Test: test\_lwgeom\_segmentize2d ... passed

Test: test\_lwgeom\_locate\_along ... passed

Test: test\_lw\_dist2d\_pt\_arc ... passed

Test: test\_lw\_dist2d\_seg\_arc ... passed

Test: test\_lw\_dist2d\_arc\_arc ... passed

Test: test\_lw\_arc\_length ... passed

Test: test\_lw\_dist2d\_pt\_ptarrayarc ... passed

Test: test\_lw\_dist2d\_ptarray\_ptarrayarc ... passed

Suite: node

Test: test\_lwgeom\_node ... passed

Suite: WKT Out Suite

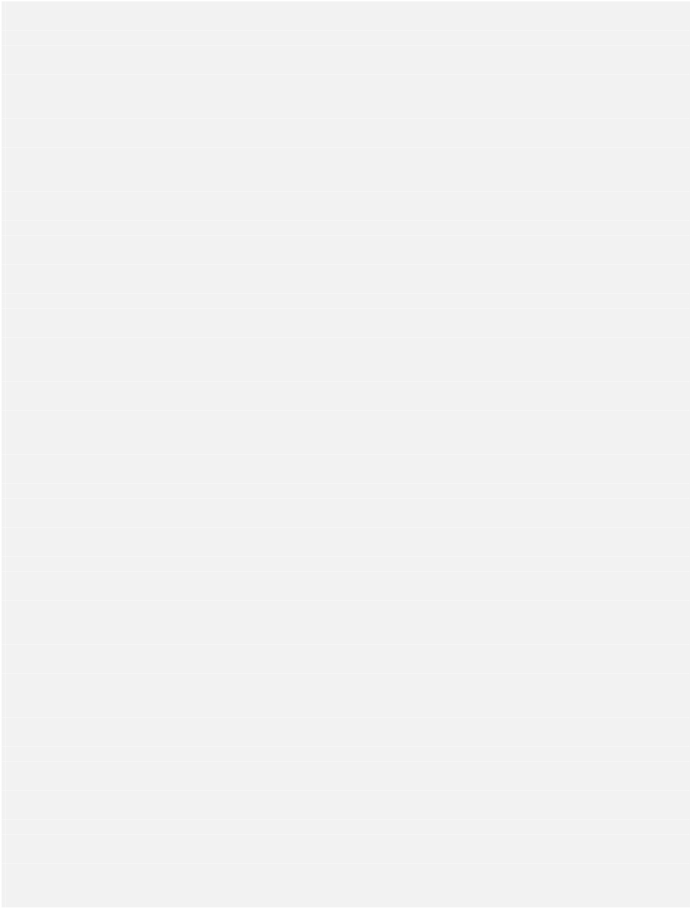
Test: test\_wkt\_out\_point ... passed

Test: test\_wkt\_out\_linestring ... passed

Test: test\_wkt\_out\_polygon ... passed

Test: test\_wkt\_out\_multipoint ... passed

Test: test\_wkt\_out\_multilinestring ... passed



13 / 686

Test: test\_wkt\_out\_multipolygon ... passed

Test: test\_wkt\_out\_collection ... passed

Test: test\_wkt\_out\_circularstring ... passed

Test: test\_wkt\_out\_compoundcurve ... passed

Test: test\_wkt\_out\_curvpolygon ... passed

Test: test\_wkt\_out\_multicurve ... passed

Test: test\_wkt\_out\_multisurface ... passed

Suite: WKT In Suite

Test: test\_wkt\_in\_point ... passed

Test: test\_wkt\_in\_linestring ... passed

Test: test\_wkt\_in\_polygon ... passed

Test: test\_wkt\_in\_multipoint ... passed

Test: test\_wkt\_in\_multilinestring ... passed

Test: test\_wkt\_in\_multipolygon ... passed

Test: test\_wkt\_in\_collection ... passed

Test: test\_wkt\_in\_circularstring ... passed

Test: test\_wkt\_in\_compoundcurve ... passed

Test: test\_wkt\_in\_curvpolygon ... passed

Test: test\_wkt\_in\_multicurve ... passed

Test: test\_wkt\_in\_multisurface ... passed

Test: test\_wkt\_in\_tin ... passed

Test: test\_wkt\_in\_polyhedralsurface ... passed

Test: test\_wkt\_in\_errlocation ... passed

Suite: WKB Out Suite

Test: test\_wkb\_out\_point ... passed

Test: test\_wkb\_out\_linestring ... passed

Test: test\_wkb\_out\_polygon ... passed

Test: test\_wkb\_out\_multipoint ... passed

Test: test\_wkb\_out\_multilinestring ... passed

Test: test\_wkb\_out\_multipolygon ... passed

Test: test\_wkb\_out\_collection ... passed

Test: test\_wkb\_out\_circularstring ... passed

Test: test\_wkb\_out\_compoundcurve ... passed

Test: test\_wkb\_out\_curvpolygon ... passed

Test: test\_wkb\_out\_multicurve ... passed

Test: test\_wkb\_out\_multisurface ... passed

Test: test\_wkb\_out\_polyhedralsurface ... passed

:

Suite: Geodetic Suite

Test: test\_sphere\_direction ... passed

Test: test\_sphere\_project ... passed

Test: test\_lwgeom\_area\_sphere ... passed

Test: test\_signum ... passed

Test: test\_gbox\_from\_spherical\_coordinates ... passed

:

Test: test\_geos\_noop ... passed

Suite: Internal Spatial Trees

Test: test\_tree\_circ\_create ... passed

Test: test\_tree\_circ\_pip ... passed

Test: test\_tree\_circ\_pip2 ... passed

Test: test\_tree\_circ\_distance ... passed

Suite: triangulate

Test: test\_lwgeom\_delaunay\_triangulation ... passed

Suite: stringbuffer

Test: test\_stringbuffer\_append ... passed

Test: test\_stringbuffer\_aprintf ... passed

Suite: surface

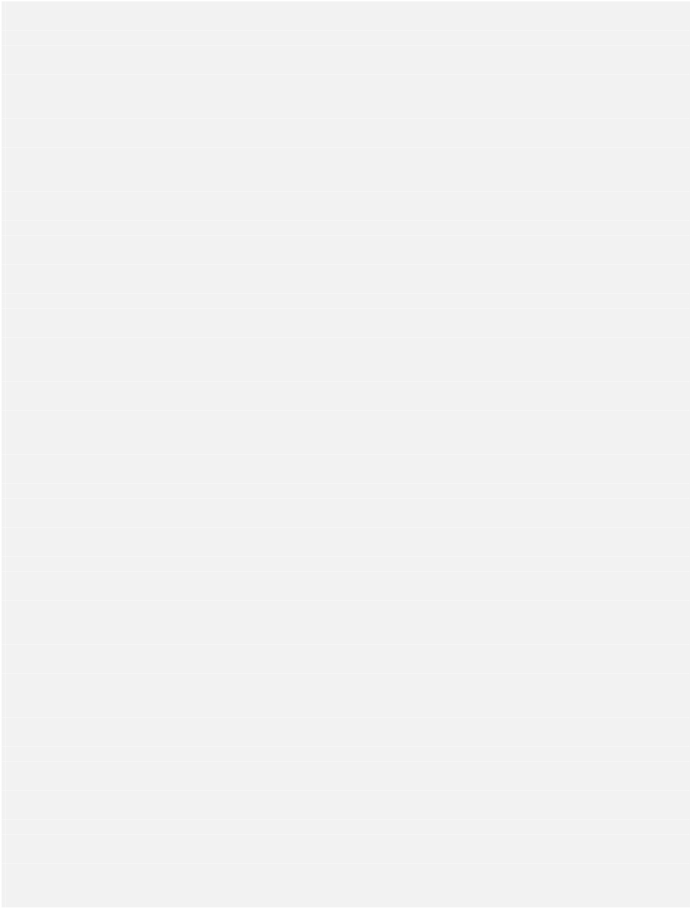
Test: triangle\_parse ... passed

Test: tin\_parse ... passed

Test: polyhedralsurface\_parse ... passed

Test: surface\_dimension ... passed

Suite: homogenize



14 / 686

Test: test\_coll\_point ... passed

Test: test\_coll\_line ... passed

Test: test\_coll\_poly ... passed

Test: test\_coll\_coll ... passed

Test: test\_geom ... passed

Test: test\_coll\_curve ... passed

Suite: force\_sfs

Test: test\_sfs\_11 ... passed

Test: test\_sfs\_12 ... passed

Test: test\_sqlmm ... passed

Suite: out\_gml

Test: out\_gml\_test\_precision ... passed

Test: out\_gml\_test\_srid ... passed

Test: out\_gml\_test\_dims ... passed

Test: out\_gml\_test\_geodetic ... passed

Test: out\_gml\_test\_geoms ... passed

Test: out\_gml\_test\_geoms\_prefix ... passed

Test: out\_gml\_test\_geoms\_nodims ... passed

Test: out\_gml2\_extent ... passed

Test: out\_gml3\_extent ... passed

Suite: KML Out Suite

Test: out\_kml\_test\_precision ... passed

Test: out\_kml\_test\_dims ... passed

Test: out\_kml\_test\_geoms ... passed

Test: out\_kml\_test\_prefix ... passed

Suite: GeoJson Out Suite

Test: out\_geojson\_test\_precision ... passed

Test: out\_geojson\_test\_dims ... passed

Test: out\_geojson\_test\_srid ... passed

Test: out\_geojson\_test\_bbox ... passed

Test: out\_geojson\_test\_geoms ... passed

Suite: SVG Out Suite

Test: out\_svg\_test\_precision ... passed

Test: out\_svg\_test\_dims ... passed

Test: out\_svg\_test\_relative ... passed

Test: out\_svg\_test\_geoms ... passed

Test: out\_svg\_test\_srid ... passed

Suite: X3D Out Suite

Test: out\_x3d3\_test\_precision ... passed

Test: out\_x3d3\_test\_geoms ... passed

--Run Summary: Type

suites

tests

asserts

Total

27

198

1728

Ran

27

198

1728

Passed

n/a

198

1728

Failed

0

0

0

Creating database ‘postgis\_reg’

Loading PostGIS into ‘postgis\_reg’

PostgreSQL 9.3beta1 on x86\_64-unknown-linux-gnu, compiled by gcc (Debian 4.4.5-8) 4.4.5,

64-bit

Postgis 2.1.0SVN - r11415 - 2013-05-11 02:48:21

GEOS: 3.4.0dev-CAPI-1.8.0 r3797

PROJ: Rel. 4.7.1, 23 September 2009

Running tests

loader/Point .............. ok

loader/PointM .............. ok

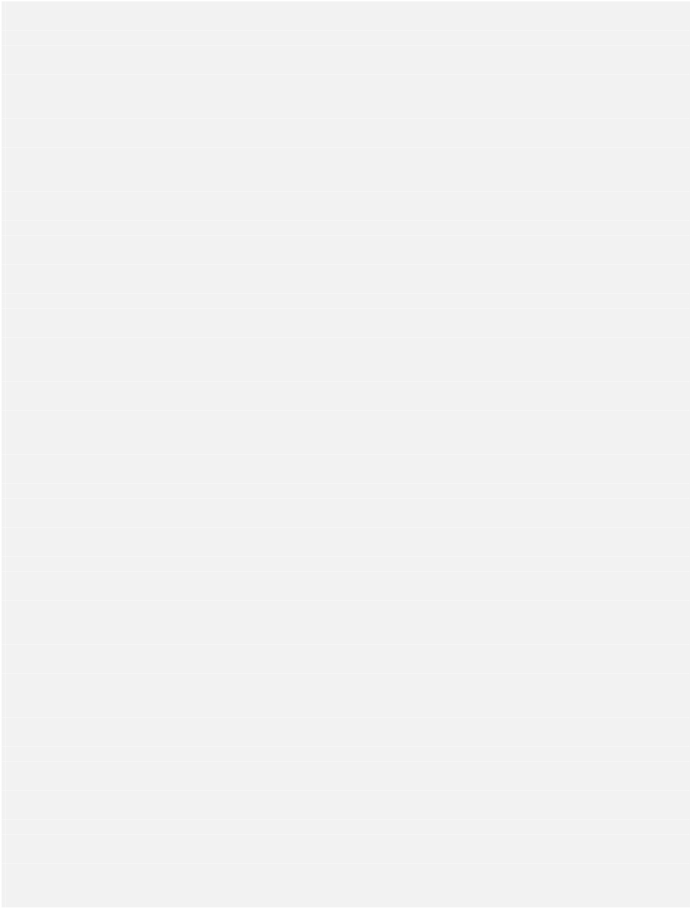
loader/PointZ .............. ok

loader/MultiPoint .............. ok

loader/MultiPointM .............. ok

loader/MultiPointZ .............. ok

←



15 / 686

loader/Arc .............. ok

loader/ArcM .............. ok

loader/ArcZ .............. ok

loader/Polygon .............. ok

loader/PolygonM .............. ok

loader/PolygonZ .............. ok

loader/TSTPolygon ......... ok

loader/TSIPolygon ......... ok

loader/TSTIPolygon ......... ok

loader/PointWithSchema ..... ok

loader/NoTransPoint ......... ok

loader/NotReallyMultiPoint ......... ok

loader/MultiToSinglePoint ......... ok

loader/ReprojectPts ........ ok

loader/ReprojectPtsGeog ........ ok

loader/Latin1 .... ok

binary .. ok

regress .. ok

regress\_index .. ok

regress\_index\_nulls .. ok

regress\_selectivity .. ok

lwgeom\_regress .. ok

regress\_lrs .. ok

removepoint .. ok

setpoint .. ok

simplify .. ok

snaptogrid .. ok

summary .. ok

affine .. ok

empty .. ok

measures .. ok

legacy .. ok

long\_xact .. ok

ctors .. ok

sql-mm-serialize .. ok

sql-mm-circularstring .. ok

sql-mm-compoundcurve .. ok

sql-mm-curvepoly .. ok

sql-mm-general .. ok

sql-mm-multicurve .. ok

sql-mm-multisurface .. ok

polyhedralsurface .. ok

polygonize .. ok

postgis\_type\_name .. ok

geography .. ok

out\_geometry .. ok

out\_geography .. ok

in\_geohash .. ok

in\_gml .. ok

in\_kml .. ok

iscollection .. ok

regress\_ogc .. ok

regress\_ogc\_cover .. ok

regress\_ogc\_prep .. ok

regress\_bdpoly .. ok

regress\_proj .. ok

regress\_management .. ok

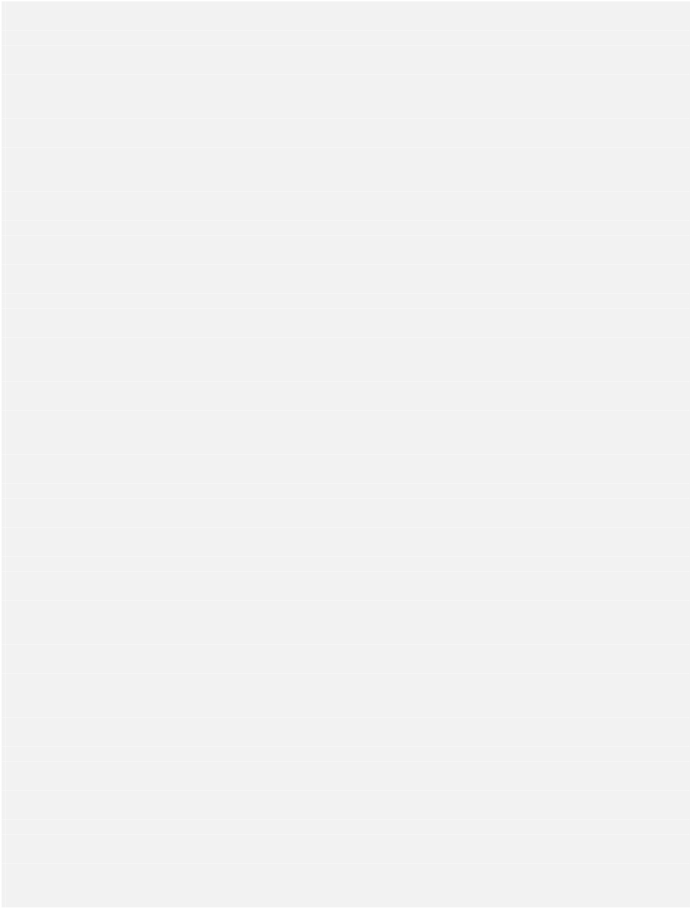
dump .. ok

dumppoints .. ok

boundary .. ok

wmsservers .. ok

wkt .. ok



16 / 686

wkb .. ok

tickets .. ok

typmod .. ok

remove\_repeated\_points .. ok

split .. ok

relate .. ok

bestsrid .. ok

concave\_hull .. ok

hausdorff .. ok

regress\_buffer\_params .. ok

offsetcurve .. ok

relatematch .. ok

isvaliddetail .. ok

sharedpaths .. ok

snap .. ok

node .. ok

unaryunion .. ok

clean .. ok

relate\_bnr .. ok

delaunaytriangles .. ok

in\_geojson .. ok

uninstall .. ok (4112)

Run tests: 90

**2.4.5**

**安装**

要安装PostGIS，输入

make install

这会将PostGIS安装文件拷贝到参数—prefix配置参数目录下的子目录

特别地：

· load和dump二进制程序安装在目录[prefix]/bin 下面

·SQL文件，例如postgis.sql, 安装在[prefix]/share/contrib下面

·PostGIS库安装在 [prefix]/lib目录下

如果你之前运行了make comments命令来生成SQL文件postgis\_comments.sql, raster\_comments.sql，使用如下命令来安装SQL文件

make comments-install

**注意**

postgis\_comments.sql, raster\_comments.sql, topology\_comments.sql 这几个脚本与典型的安装方式是区别开的，因为他们依赖程序

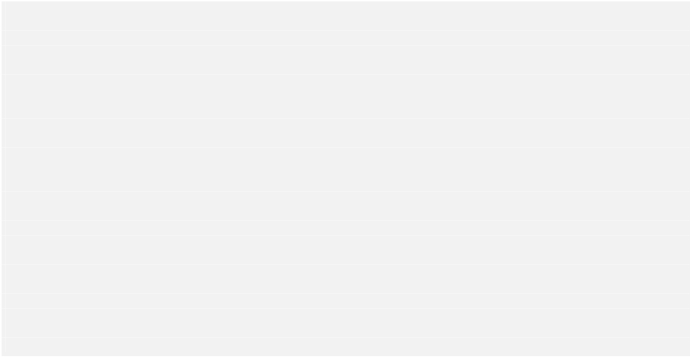
**xsltproc**.

**2.5**

**在低于PostgreSQL 9.1版本以下创建一个有空间功能支持的数据库**

创建PostGIS数据库的第一步是创建一个PostgreSQL数据库：

createdb [yourdatabase]





17 / 686

许多PostGIS函数是用PL/pgSQL存储过程写的。因此，创建PostGIS数据库之前的下一步是创建PL/pgSQL语言支持。这个可以通过如下命令实现。对于PostgreSQL 8.4+版本，该语言已经默认安装了

createlang plpgsql [yourdatabase]

现在加载你的PostGIS 数据库对象和函数到数据库中，办法是执行postgis.sql 脚本文件(该文件在 [prefix]/share/contrib 目录下，是在配置步骤中指定的).

psql -d [yourdatabase] -f postgis.sql

为了得到一个完整的EPSG坐标系的完备记录，你需要执行脚本spatial\_ref\_sys.sql 来插入数据到spatial\_ref\_sys 表中。有了这些数据，你可以对几何对象数据使用函数ST\_Transform() 来操作

psql -d [yourdatabase] -f spatial\_ref\_sys.sql

如果你希望对PostGIS函数添加注释，那就执行postgis\_comments.sql

这些函数注释可以通过在psql中输入命令\dd [function\_name] 来查看

psql -d [yourdatabase] -f postgis\_comments.sql

安装栅格功能支持

psql -d [yourdatabase] -f rtpostgis.sql

安装栅格功能支持的注释。当你使用psql或PgAdmin或者其他任何PostgreSQL工具时候，该注释功能会向你介绍每个栅格函数的快速的帮助

psql -d [yourdatabase] -f raster\_comments.sql

安装你的拓扑功能支持

psql -d [yourdatabase] -f topology/topology.sql

安装拓扑功能注释。当你使用psql或PgAdmin或者其他任何PostgreSQL工具时候，该注释功能会向你介绍每个拓扑函数和拓扑类型的快速的帮助

psql -d [yourdatabase] -f topology/topology\_comments.sql

如果你把之前备份的数据恢复到新的数据库中，执行命令：

psql -d [yourdatabase] -f legacy.sql

**注意**

还有一种备选方案就是将legacy\_minimal.sql 安装在裸机上面用于恢复表，和应用一起使用，就像使用

MapServer 和 GeoServer一样。如果你有使用距离或长度计算的视图，你需要完整的legacy.sql

你可以在你恢复和清理垃圾后，再执行uninstall\_legacy.sql 来规避一些不推荐的函数

**2.6**

**使用EXTENSIONS插件方式创建空间数据库**

如果你使用PostgreSQL 9.1+ 数据库，并且编译并安装了postgis扩展，你可以以新的方式来创建空间数据库

createdb [yourdatabase]

核心的PostGIS扩展插件安装PostGIS geometry, geography, raster, spatial\_ref\_sys 功能及表和所有的函数以及函数的注释，只需要简单执行：

CREATE EXTENSION postgis;



18 / 686

psql -d [yourdatabase] -c "CREATE EXTENSION postgis;"

拓扑功能是一个单独的扩展，需要执行如下命令：

psql -d [yourdatabase] -c "CREATE EXTENSION postgis\_topology;"

如果你想要把之前备份的数据导入到新的数据库中，执行：

psql -d [yourdatabase] -f legacy.sql

你可以在你恢复和清理垃圾后，再执行uninstall\_legacy.sql 来规避一些不推荐的函数

**2.7**

**安装，升级Tiger Geocoder 地理信息编码器和装载数据**

PostGIS额外的东西，比如Tiger 地理编码器可能没有打包在你的PostGIS发布版本中，但是在postgis2.2.0dev.tar.gz文件中是包含的。这里的说明同样在目录 extras/tiger\_geocoder/tiger\_2011/README中有说明

如果你的系统是Windows系统，那么你没有tar命令程序（Linux打包和解压工具），你可以使用http://www.7-zip.org/ 提供的解压软件来解压你的tar包

**2.7.1**

**使用扩展来让你的PostGIS数据库支持Tiger Geocoder插件**

如果你使用PostgreSQL 9.1+ 和 PostGIS 2.1.0,你可以充分利用新的创建扩展的方式来安装tiger geocoder插件：

1. 首先获取二进制版本的PostGIS 2.1.0 或者已经编译好的软件包，然后正常安装。这会为tiger geocoder 安装必要的插件

2. 通过psql或pgAdmin或任何其他工具连接数据库，然后执行下面的命令。注意：如果你的数据库已经安装了PostGIS了，那么第一步可以省略。如果你也安装了fuzzystrmatch ，那么第二步也不需要做

CREATE EXTENSION postgis;

CREATE EXTENSION fuzzystrmatch;

CREATE EXTENSION postgis\_tiger\_geocoder;

3. 在数据库中执行下面的SQL来确认你是否正确安装了postgis\_tiger\_geocoder：

SELECT na.address, na.streetname,na.streettypeabbrev, na.zip

FROM normalize\_address(‘1 Devonshire Place, Boston, MA 02109’) AS na;

输出应该是：

address | streetname | streettypeabbrev |

zip

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

1 | Devonshire | Pl

| 02109

4.根据你在tiger\_loader\_2012.sql文件中配置的服务器执行路径，在表tiger.loader\_platform中新增一条记录，例如创建一条遵从sh方式的配置如下

INSERT INTO tiger.loader\_platform(os, declare\_sect, pgbin, wget, unzip\_command, psql,

path\_sep,

loader, environ\_set\_command, county\_process\_command)

SELECT ‘debbie’, declare\_sect, pgbin, wget, unzip\_command, psql, path\_sep,

loader, environ\_set\_command, county\_process\_command

FROM tiger.loader\_platform

WHERE os = ‘sh’;

←

19 / 686

然后编辑declare\_sect列的一些执行脚本路径参数，该参数应该和Debbie的pg, unzip,shp2pgsql, psql 等路径相匹配。

如果你不编辑这个文件或者表loader\_platform，那么该扩展只会包含默认的路径配置等参数，那样的话你需要在执行函数Loader\_Generate\_Nation\_Script 和 Loader\_Generate\_Script 完成后，再编辑生成的脚本。

5. 然后运行Loader\_Generate\_Nation\_Script 和 Loader\_Generate\_Script 函数来确保是使用你配置的名称。即

SELECT Loader\_Generate\_Nation\_Script(‘debbie’);

**2.7.1.1**

**把Tiger Geocoder常规安装方式转变成扩展方式安装**

如果你没有使用扩展方式来安装tiger geocoder 插件，你可以通过下面的步骤来转变成扩展方式安装：

1. 参考2.7.4 节来进行非扩展方式升级for the non-extension model upgrade.

2. 通过psql或者pgAdmin工具连接数据库，然后执行如下命令：

CREATE EXTENSION postgis\_tiger\_geocoder FROM unpackaged;

**2.7.1.2**

**安装和使用PAGC地址标准化工具**

广大用户经常抱怨的一个问题是地址正则化函数Normalize\_Address ，一个编码前先进行正则化的函数。该正则化函数远没有想象的完美，目前我们也花了大量资源来让它变得完美。本身的话，我们也正在和另一个项目做集成，该项目有一个更好的地址标准化引擎。该项目目前是一个独立的项目，是PAGC的子项目。该项目支持的PostgreSQL标准化工具扩展的源码在

PAGC PostgreSQL Address Standardizer。要使用这个标准化工具，你可以编译PAGC扩展，然后在数据库中安装该扩展

特别地，PAGC项目和标准化工具依赖于PCRE程序，该程序目前按照在大多数类Unix系统（包括Linux）上面，但你也可以下载最新的源码: http://www.pcre.org。该程序需要Perl的Regexp::Assemble 模块支持

对于Windows用户，PostGIS 2.1+包已经包含了address\_standardizer插件，因此不用编译，只需要直接使用命令CREATE EXTENSION 来创建该扩展。

按照Perl的模块 Regex::Assemble方法

cpan Regexp::Assemble

如果的系统是Ubuntu / Debian ，你可能需要下面的办法

sudo perl -MCPAN -e "install Regexp::Assemble"

编译

svn co svn://svn.code.sf.net/p/pagc/code/branches/sew-refactor/postgresql

address\_standardizer

cd address\_standardizer

make

#if you have in non-standard location pcre try

←

# make SHLIB\_LINK="-L/path/pcre/lib -lpostgres -lpgport -lpcre" CPPFLAGS="-I.

/include"

make install

安装成功后，你需要连接数据库并执行SQL:

CREATE EXTENSION address\_standardizer;

-I/path/pcre ←

一旦你在数据库中像安装postgis\_tiger\_geocoder一样安装了address\_standardizer插件，那么可以用函数Pagc\_Normalize\_Ad

来替代函数Normalize\_Address。该扩展的另一个好的地方是tiger无关的，因此该扩展可用用别的数据源比如国际化地址.

20 / 686

**2.7.2**

**不使用扩展安装的方式来对PostGIS数据库安装Tiger Geocoder功能支持**

首先使用先前的指导安装PostGIS

如果你没有extras目录，下载<http://postgis.net/stuff/postgis-2.2.0dev.tar.gz>

然后

tar xvfz postgis-2.2.0dev.tar.gz

cd postgis-2.2.0dev/extras/tiger\_geocoder/tiger\_2011

编辑tiger\_loader\_2012.sql 文件，把path（译者注：该参数配置在这个文件的第78行）设置成你在服务器上执行的路径或者另一种方式你可以在安装该插件后，更新loader\_platform表。如果你不编辑这个文件或者表loader\_platform，那么该扩展只会包含默认的路径配置等参数，那样的话你需要在执行函数Loader\_Generate\_Nation\_Script 和 Loader\_Generate\_Script 完成后，再编辑生成的脚本。

如果你第一次安装Tiger geocoder 扩展，那么要么编辑create\_geocode.bat 脚本（如果你是在Windows上面安装PostgreSQL），要么编辑

create\_geocode.sh 脚本（如果你在Linux/Unix/Mac OSX上面安装PostgreSQL）。然后使用命令行运行相应的脚本。

要确认一下你的数据库中有了一个名叫tiger的schema，并且也在你的search\_path参数里面，如果没有在search\_path参数里面，使用下面的命令添加

ALTER DATABASE geocoder SET search\_path=public, tiger;

标准化地址函数除了复杂的、奇怪的地址之外，无论数据多少都一样工作。执行下面的测试来验证是否正常工作：

SELECT pprint\_addy(normalize\_address(‘202 East Fremont Street, Las Vegas, Nevada 89101’))

As pretty\_address;

pretty\_address

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

202 E Fremont St, Las Vegas, NV 89101

←

**2.7.3**

**加载 Tiger 数据**

关于使用该扩展加载数据可以参考extras/tiger\_geocoder/tiger\_2011/README获取更详细的步骤，这里只是一般步骤。

load数据的过程是：从census 网站上面下载各自的国家数据文件，州数据文件，然后从这些数据文件中抽取数据，再把每个州的数据load到各自的州表中。每一个州信息表继承了tiger schema的表，这样的话查询表数据的时候就可以从这些子表中查询所有数据，或者可以在任何时候比如当你需要重新load 每个州的信息数据或者不再需要这些表的时候，都可以使用Drop\_State\_Tables\_Generate\_Script 来drop掉这些州信息（state）子表。

为了能够load数据，你需要下面的工具：

· 一个能解压从census 网站下载的zip文件的解压工具

对于类Unix系统：大部分类Unix平台都安装了unzip可执行命令，可以用于解压

对于Windows平台：7-zip软件是一个免费的压缩和解压工具，你可以从<http://www.7-zip.org/> 下载

· shp2pgsql ：命令行工具，默认在你安装PostGIS时候就安装了

· wget ：一个web抓取工具，在大多数Unix/Linux系统上面都安装了

如果你在Windows系统上，你可以下载预编译的二进制程序软件<http://gnuwin32.sourceforge.net/packages/wget.htm>

如果你从tiger\_2010版本升级，你需要首先生成并运行Drop\_Nation\_Tables\_Generate\_Script函数生成的SQL脚本。在你load任何州信息数据到数据库之前，你需要先load国家信息数据，该数据可以通过函数Loader\_Generate\_Nation\_Script生成，该函数会为你生成一个load脚本。函数. Loader\_Generate\_Nation\_Script 从2010版本起，生成的脚本，需要在升级或首次安装时候一步执行到位。

要load 州信息数据，参考Loader\_Generate\_Script 函数，生成一个数据加载脚本到任何你需要的平台上。注意：你可以一步一步来load 州数据，你不必一次性将所有州信息数据都安装。你可以根据需要来load数据。

当你所需要的州信息数据完全load进去后，记住要运行：

21 / 686

SELECT install\_missing\_indexes();

关于此参考Install\_Missing\_Indexes

测试地址是否正常编码，请使用函数Geocode 来做一个地址编码

**2.7.4**

**升级你安装的Tiger Geocoder**

如果你已经在你的PostGIS 2.0+版本中安装了Tiger Geocoder 扩展，你可以在任何时候升级该扩展的函数，即便是一个临时性的PostGIS tar包版本（该版本可能有一些你需要的必须修复的补丁）。下面的只适用于没有安装扩展的Tiger geocoder 。

如果你没有extras目录，下载<http://postgis.net/stuff/postgis-2.2.0dev.tar.gz>

tar xvfz postgis-2.2.0dev.tar.gz

cd postgis-2.2.0dev/extras/tiger\_geocoder/tiger\_2011

如果是Windows平台，找到upgrade\_geocoder.bat 脚本，如果是Linux/Unix/Mac OSX平台，找到upgrade\_geocoder.sh脚本。编辑这些文件让他们在执行相关命令时候可以通过权限认证（比如加上账号和密码）

如果你从2010 或 2011版本的Tiger Geocoder扩展升级，确保你取消了“${PSQL\_CMD} -d "${THEDB}" -f "upgrade\_geocode.sql"（译者注：这一行是upgrade\_geocoder.sh脚本里面的） 这一行注释，以便你能获取到最新的脚本来加载2012版本的数据然后在命令行上面运行相应的脚本：

然后删除掉所有的国家相关表，然后加载新的数据，生成DROP脚本请参考Drop\_Nation\_Tables\_G

SELECT drop\_nation\_tables\_generate\_script();

运行生成删除数据的SQL脚本

使用SELECT语句生成国家数据load脚本，参考函数Loader\_Generate\_Nation\_Script

对于Windows版本

SELECT loader\_generate\_nation\_script(‘windows’);

对于unix/linux版本

SELECT loader\_generate\_nation\_script(‘sh’);

参考章节 2.7.3 获取怎样生成脚本的指导。上面的语句只需要执行一次。

**注意**

如果你数据库中包含的是2010和2011 混合州数据，你可以一个一个州表升级。在你升级一个州数据到2011版本之前，你先drop掉2010版本的表，drop脚本使用函数Drop\_State\_Tables\_Generate\_Script生成

**2.8**

**从一个template模板数据库中创建一个支持空间功能的数据库**

有一些PostGIS版本（特别是Win32平台上的PostGIS>=1.1.5的安装工具安装的版本）会将PostGIS函数加载到一个叫做template\_postgis的临时数据库中。如果你的PostgreSQL数据库中已经存在了这个数据库，那么对于用户或者应用来说，通过一个简单明了就可以创建一个有空间支持的数据库。注意：无论哪种情况，数据库用户都必须有创建新数据库的授权

shell命令执行方式:

# createdb -T template\_postgis my\_spatial\_db

SQL语句执行方式:

postgres=# CREATE DATABASE my\_spatial\_db TEMPLATE=template\_postgis



22 / 686

**2.9**

**升级**

升级已有的空间数据库可以认为是替换或引入新的PostGIS数据库对象定义

不幸的是，不是所有的数据库对象定义可以在一个在线数据库上面轻易地替换掉，因此有时候最好的办法是使用dump/reload的方式

PostGIS 提供了对于小版本或bug修复版本的软升级方式，和对于大版本的硬升级方式

在升级PostGIS之前，一定要备份你的数据。如果你在使用pg\_dump命令带有-Fc参数的方式备份数据，硬升级后可以恢复备份的数据

**2.9.1**

**软升级**

如果你通过扩展方式安装PostGIS,你同样可以通过扩展方式来升级。如果你通过老的脚本方式来安装，那么你应该通过脚本方式来升级。请参考相应章节。

**2.9.1.1**

**9.1前版本软升级或不使用扩展方式升级**

这一节主要适用于那些没有使用扩展模式安装的情形。如果你的数据库有扩展，使用扩展升级时候得到信息

can’t drop ... because postgis extension depends on it

编译安装后 (make install)，你应该在安装目录中找到postgis\_upgrade.sql 和 rtpostgis\_upgrade.sql。例如/usr/share/postgresql/9.3/contrib/postgis\_upgrade.sql。执行脚本the postgis\_upgrade.sql。如果你已经安装了栅格功能安装了，你同样需要执行脚本/usr/share/postgresql/9.3/contrib/postgis\_upgrade.sql。如果你需要从PostGIS 1.\* 版本升级到 PostGIS 2.\* 版本，或者从PostGIS 2.\* 之前的版本升级到r7409版本，你需要通过硬升级方式升级

psql -f postgis\_upgrade.sql -d your\_spatial\_database

同样对于栅格和拓扑扩展，也需要上面的执行步骤。升级文件分别是rtpostgis\_upgrade\*.sql 和topology\_upgrade\*.sql 。如果你需要这些功能升级，执行：

psql -f rtpostgis\_upgrade.sql -d your\_spatial\_database

psql -f topology\_upgrade.sql -d your\_spatial\_database

**注意**

如果你没有找到文件postgis\_upgrade\*.sql 来升级到你的版本，你可能使用的版本太老了，以至于无法使用软升级方式，这样你必须要硬升级

函数PostGIS\_Full\_Version 会以"procs need upgrade"这样的信息，告诉你这种类型升级的条件

**2.9.1.2**

**使用扩展方式升级9.1+版本**

如果你原本就以扩展方式安装了PostGIS，那么你同样也可以用扩展方式来升级。使用扩展来升级小版本是相当无痛苦的

ALTER EXTENSION postgis UPDATE TO "2.2.0dev";

ALTER EXTENSION postgis\_topology UPDATE TO "2.2.0dev";



23 / 686

如果你得到一个错误信息如下：

No migration path defined for ... to 2.2.0dev

那么你需要备份数据库，如第2.6节所述创建一个新的一个库，然后在这个新的数据库恢复数据

如果你得到这样的通知消息：

“Version "2.2.0dev" of extension "postgis" is already installed”

译者备注：这个表示postgis扩展已经安装

那么到目前为止所有的都已经安装好了，你可以无视这个信息。除非你想尝试从SVN版本升级到下一个版本（而这个版本没有新的版本号），这种情况下，你可以在版本字符串上面加上“next“字符串，下次升级时候你还是需要把”next“后缀删掉

ALTER EXTENSION postgis UPDATE TO "2.2.0devnext";

ALTER EXTENSION postgis\_topology UPDATE TO "2.2.0devnext";

**注意**

如果你已经安装了一个没有具体版本的PostGIS，你可以在恢复数据前直接跳过PostGIS扩展方式的安装，因为备份只包含了

CREATE EXTENSION postgis ，这样的话直接选一个最新的PostGIS版本好了

**2.9.2**

**硬升级**

硬升级是指我们需要全部dump出数据，然后再把数据重新load到数据库中。当PostGIS对象的内部存储改变或者当软升级不可用的时候，你需要硬升级。在每个版本发布记录（Release Notes）的附录章节，你可以看到你是否需要硬升级的方式来进行升级

dump和reload 程序都需要PostGIS的Perl程序postgis\_restore.pl脚本支持，该脚本会处理属于PostGIS的数据库对象定义（包括旧版本的），以及让你在恢复schema和数据到一个已经安装PostGIS的数据库时候，跳过数据重复的错误和不推荐的数据库对象

关于Windows用户硬升级的补充说明，请参考 [Windows Hard upgrade.](http://trac.osgeo.org/postgis/wiki/UsersWikiWinUpgrade)

过程如下：

1. 创建一个自定义的你要升级的数据库（暂时称之为olddb）dump命令，包含二进制blob（-b）对象和详细日志输出（-v）。用户只需要是这个数据库的所有者即可，不必是postgres超级管理员账户

pg\_dump -h localhost -p 5432 -U postgres -Fc -b -v -f "/somepath/olddb.backup" olddb

2. 在新的数据库中安装一个新的PostGIS库——我们会把这个数据库称作newdb。请参考2.5 节和2.6 节获取如何这样做的信息。

在你dump文件中，数据表spatial\_ref\_sys会被恢复，但是他们不会覆盖已经存在的spatial\_ref\_sys表。

这是为了确保官网的bug修复可以以正确的方式影响到被恢复的数据库中。如果因为任何原因，你真的想覆盖掉已经存在的表入口，创建新数据库时候不执行脚本spatial\_ref\_sys.sql。如果你的数据库真的很老或者你知道你已经在你的视图或函数中使用了已经很久不推荐的函数，你可能需要执行脚本legacy.sql 来恢复这些老版本的函数和视图。除非真的必要，你才可以这样做。如果可能的话，在dump数据库数据前，请优先考虑升级你的视图和函数，不推荐的函数会稍后被脚本uninstall\_legacy.sql所删除。

3.使用脚本postgis\_restore.pl把你备份的数据恢复到新创建的数据库newdb中。如果有不可预料的错误，错误将会输出到psql的标准输出中，最好把错误重定向到一个错误文件中

perl utils/postgis\_restore.pl "/somepath/olddb.backup" | psql -h localhost -p 5432 -U

postgres newdb 2> errors.txt

←



PostGIS 2.2.0dev 开发手册

24 / 686

下面的情况可能会引起一些错误

1. 你的视图或者函数里面可能使用了一些已经不推荐的PostGIS 对象。为了解决这个问题，兼容以前的版本，你需要执行脚本legacy.sql 来进行恢复，或者需要恢复到一个还包含这些对象的PostGIS版本，然后进行代码移植。如果脚本legacy.sql 能解决你的问题，别忘了修复你的代码，停止使用这些不推荐的函数，然后通过执行uninstall\_legacy.sql脚本来抛掉这些不推荐的数据库对象。

2.dump出的表spatial\_ref\_sys中含有一些用户自定义的记录，其SRID值是无效。有效的SRID值应该是大于0，并且小于

999000的。SRID值大于999000小于999999范围的值是保留内部使用的，而值大于999999是完全不能被使用的。你所有的无效的SRID记录都会被保留，而大于999999的会被移动到保留范围内的值，但spatial\_ref\_sys表有约束，会遵守primary key条件，将大于999999的插入更新到保留的范围内（译者注：相当于upsert方式）( 当多个无效的SRID值被转换成相同的保留的SRID值).

为了修复这个问题，你应该把你自定义的SRS表的SRID值都变成有效的（可能在910000与910999之间），然后把用到你自定义的SRID的所有相关表都转换成新的SRID值（参考UpdateGeometrySRID），然后删除掉无效的SRID记录并对表spatial\_ref\_sys添加如下约束：

ALTER TABLE spatial\_ref\_sys ADD CONSTRAINT spatial\_ref\_sys\_srid\_check check (srid > 0

AND srid < 999000 );

ALTER TABLE spatial\_ref\_sys ADD PRIMARY KEY(srid));

译者注：所以最根本的办法是自定义SRID值时候一定不要超过PostGIS 保留范围内的SRID值。

←

**2.10**

**安装过程中的常见问题**

当你安装或者升级时候没有得到你预想的结果时候，需要做如下几个步骤：

1．检查是否已安装PostgreSQL 9.1或更新版本，并且你正在编译的PostgreSQL源码和你正在运行的PostgreSQL版本是一致的。当你的(linux)环境已经安装了PostgreSQL，或者你之前用别的方式安装了postgresql，但却忘记了。

PostGIS只会PostgreSQL 9.1或更新版本运行。如果使用更老的版本，就会报一下奇怪的、意想不到的错误信息。你需要

检查PostgreSQL正在运行的版本，使用psql连接到数据库并运行此查询：

SELECT version();

如果您正在运行一个RPM版本postgresql环境，你可以使用如下rpm命令检测之前的安装包信息：

命令如下：rpm -qa| grep postgresql

2．如果你升级失败，确保进入一个恢复并安装PostGIS的数据库

select postgis\_full\_version()；

同时检查configure命令正确地检测到了PostgreSQL的位置和版本信息以及Proj4库和GEOS库。

1.configure命令的输出用于生成postgis\_config.h文件。检查这个文件里面POSTGIS\_PGSQL\_VER-SION, POSTGIS\_PROJ\_VERSION and POSTGIS\_GEOS\_VERSION变量被正确设置。

译者注：configure命令执行前，需要设置好环境变量，并source ~/pg安装目录的.bashrc或者.bash\_profle文件，具体看你把环境变量配置哪个文件中。

例如：export LD\_LIBRARY\_PATH=/opt/gdal-1.9.1/lib:/opt/geos-3.3.4/lib:/opt/json-c-0.9/lib:/opt/proj-4.8.0/lib:$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:$LD\_LIBRARY\_PATH

**2.11**

**JDBC插件扩展**

JDBC插件扩展提供了java对象对应的内部定义的PostGIS类型。这些对象可以用来java写查询PostGIS数据库以及图像绘制或GIS数据计算的Java客户端工具.

译者注：就是把postgis.jar包打包到你的Java库中，否则你的Java程序无法识别geometry这样的对象类型。

1．进入PostGIS安装环境的Java安装目录的子目录

2．运行ant命令。复制postgis.jar包到任意你保存Java库的目录

PostGIS 2.2.0dev 开发手册

25 / 686

构建JDBC扩展插件过程中需要在当前环境变量$CLASSPATH对应的目录中有一个PostgreSQL数据库的JDBC驱动程序。如果PostgreSQL的JDBC驱动程序位于其他地方，你可以单独使用-D参数指定的JDBC驱动的jar包位置如下：

# ant -Dclasspath=/path/to/postgresql-jdbc.jar

PostgreSQL JDBC 驱动下载地址 http://jdbc.postgresql.org

**2.12**

**批量加载/批量转储工具**

译者注：对dba来说，load和dump这种说法更习惯一些，load是大批量的导入数据，dump是大批量导出数据的意思。

数据批量加载和批量导出工具是安装PostGIS时候被自动同时安装的，手动安装该工具步骤如下:

# cd postgis-2.2.0dev/loader

# make

# make install

加载工具指的是shp2pgsql这个命令，它把ESRI Shape格式的文件(译者注：后缀名为.shp)转换为SQL脚本，以便在PostgreSQL数据库中执行。批量导出工具指的是pgsql2shp，它把PostGIS的表或者一个查询转换成ESRI Shape格式的文件。关于更多详细文档信息，请查看在线帮助和相关手册