Blog do Albuquerque

Matérias do dia-a-dia de minha vida, quando consigo
fique atualizado via rss

Instalar PostGIS no Linux Debian

Publicado: 26/02/2012 em **Linux**, **Serviços IP**
Tags:**GIS**, **postGIS**, **PostgreSQL**

2

i
Rate This

1. Introdução
PostGIS adiciona suporte para objetos geográficos ao banco de dados objeto-relacional PostgreSQL (http://www.postgresql.org/). A primeira versão do PostGIS foi liberada em 2001 sob licença GPL (http://pt.wikipedia.org/wiki/GNU_General_Public_License).Com efeito, PostGIS "habilita espacialmente" o servidor PostgreSQL, permitindo a ele seja utilizado como um banco de dados espacial back-end (servidor de retaguarda) para sistemas de informação geográfica (SIG).

PostGIS segue o OpenGIS "Simple Features Specification for SQL" e foi certificado como compatível com os perfis de "Tipos e funções". Diversas ferramentas trabalham com PostGIS, tais como uDIG (http://udig.refractions.net/), Mapserver (http://mapserver.gis.umn.edu/), GeoTools (http://www.geotools.org/), GeoServer (http://geoserver.org/), GRASS (http://grass.itc.it/), OpenJUMP (http://www.openjump.org/), gvSIG (http://en.wikipedia.org/wiki/GvSIG) e QGIS (http://qgis.org/). E naturalmente qualquer linguagem que trabalha com PostgreSQL pode trabalhar com PostGIS — a lista inclui Perl, PHP, Python, TCL, C, C++, Java, C#, e outras.

2. Exigências
Para detalhes de instalação do PostgreSQL no Linux Debian, veja o post "Tutorial PostgreSQL (instalar em ambiente Linux – Debian) (http://wp.me/pCWn2-eJ)".

3. Instalar extensão PostGIS
Após ter instalado o PostgrSQL, realizar os seguintes procedimentos:

3.1) Verificar instalações e versões disponiveis de postgreSQL e postGIS

```
$ apt-cache policy postgresql
postgresql:
Instalado: 8.4.10-0squeeze1
Candidato: 8.4.10-0squeeze1
Tabela de versão:
*** 8.4.10-0squeeze1 0

$ apt-cache policy postgis
postgis:
Instalado: (nenhum)
Candidato: 1.5.1-5
Tabela de versão:
1.5.1-5 0
```

Com a resposta aos dois comandos acima, conclui-se que PostgreSQL está instalado na máquina alvo (versão 8.4), mas que o PostGIS não está instalado. A versão do PostGIS disponível no repositório Debian candidata à instalação é a 1.5.

3.2) Instalar o PostGIS referente a versão 8.4 do PostgreSQL

```
# apt-get install postgis postgresql-8.4-postgis
```

Onde:

1. postgis :contém binários do PostGIS em nível usuário, arquivos comuns e documentação. Sugere postgresql-8.4-postgis
2. postgresql-8.4-postgis :contém suporte a objetos geográficos para o PostgreSQL 8.4. Depende de postgis e de postgresql-8.4

Após a instalação descrita acima, o manual do PostGIS ficará depositado em `file:///usr/share/doc/postgis/postgis/postgis.html` , com mais detalhes quanto a sua instalação e configuração.

3.3) Criar perfil de grupo e usuário
Usualmente, a melhor maneira de manipular dados GIS no banco de dados PostgreSQL é usando papel e usuário diferentes do default “postgres”, que deve ser usado apenas para tarefas administrativas. No exemplo deste post, será criado o papel e usuário “gis” para a manipulação de dados no servidor PostgreSQL. Para se obter um ambiente mais seguro, podem ser criados outros usuários com diferentes direitos (SELECT, INSERT, UPDATE, DELETE) sobre as diferentes classes SIG.

Conectar ao servidor PostgreSQL (como usuário postgres), criar os papeis “gisgroup” e “gis” (se desejado, escolha menos permissões por razões de segurança), e atribuir a função de login “gis” para o papel do grupo “gisgroup”:

```
# su - postgres
postgres$ psql
psql=# CREATE ROLE gisgroup NOSUPERUSER NOINHERIT CREATEDB
NOCREATEROLE;
psql=# CREATE ROLE gis LOGIN PASSWORD 'minhasenha' NOINHERIT;
psql=# GRANT gisgroup TO gis;
```

Observe que agora o usuário “gis” já pode fazer login no servidor PostgreSQL com senha “minhasenha”. Uma maneira fácil de fazer isto é fazer login no servidor PostgreSQL através do aplicativo phpPgAdmin.

a) Verificar o que fora criado

```
psql=# \dg
                          Lista de roles
+--------------+-------------------------+-----------------------+
| Nome da role |       Atributos         |       Membro de       |
+--------------+-------------------------+-----------------------+
| gis          | Nenhuma herança         | {gisgroup}            |
| gisgroup     | Nenhuma herança         | {}                    |
|              : Cria BD                                         |
|              : Não pode efetuar login                          |
| postgres     | Super-usuário           | {}                    |
|              : Cria role                                       |
|              : Cria BD                                         |
+--------------+-------------------------+-----------------------+
```

3.4) Criar um template de base de dados PostGIS
Isto é prático, tornando fácil a criação das bases de dados GIS no servidor. Sem a criação deste template, o usuário teria que repetir todos os passos abaixo todas as vezes que necessitasse criar uma base de dados PostGIS. O template é a mesma coisa de uma base de dados normal, mas é marcada como template para não ser removida ou alterada inadvertidamente. O resultado deste comando são as funções do PostGIS carregadas dentro deste template de banco de dados:

```
# su - postgres
postgres$ psql
psql=# create database postgistemplate with template = template1
encoding = 'utf8' owner = gis;
psql=# COMMENT ON DATABASE postgistemplate IS 'postgis template
database';
psql=# UPDATE pg_database SET datistemplate=true WHERE
datname='postgistemplate';
psql=# \c postgistemplate
psql=# create language plpgsql;
psql=# \i /usr/share/postgresql/8.4/contrib/postgis-1.5/postgis.sql
psql=# \i /usr/share/postgresql/8.4/contrib/postgis-
1.5/spatial_ref_sys.sql
psql=# \q
```

Ou ainda de outra forma:

```
# su - postgres
postgres$ createdb postgistemplate -T template1 -E utf-8 -O gis
"postgis template database"
postgres$ createlang plpgsql postgistemplate
postgres$ psql -d postgistemplate -f
/usr/share/postgresql/8.4/contrib/postgis-1.5/postgis.sql
postgres$ psql -d postgistemplate -f
/usr/share/postgresql/8.4/contrib/postgis-1.5/spatial_ref_sys.sql
postgres$ psql
psql=# UPDATE pg_database SET datistemplate=true WHERE
datname='postgistemplate';
psql=# COMMENT ON DATABASE postgistemplate IS 'postgis template
database';
psql=# \q
```

Onde:

1. createdb :cria uma nova base de dados PostgreSQL. O usuário que executa este comando se torna o proprietário desta nova base de dados (no caso “postgres”).
2. createlang :PostGIS requer a extensão de linguagem procedural PL/pgSQL, pois muitas das funções PostGIS são escritas nesta linguagem. Para habilitar a PL/pgSQL, deve-se usar este comando “createlang”. É um utilitário que adiciona uma nova linguagem de programação para um banco de dados PostgreSQL. É apenas um invólucro em torno do comando CREATE LANGUAGE.
3. createlang -d :banco de dados para instalar linguagem psql.
4. psql :trata-se de um terminal “front-end” interativo para o PostgreSQL. Ele permite que se digite os comandos interativamente, enviá-los para o servidor PostgreSQL e ver os resultados dos comandos.
5. psql -d :especifica o nome do banco de dados para conectar-se.
6. psql -f :usa o arquivo descrito após a chave como a fonte de comandos, em vez de ler comandos interativamente.
7. postgis.sql :arquivo com definições de funções e objetos do PostGIS.
8. spatial_ref_sys.sql :arquivo de definições utilizado para um sistema completo de identificadores de definições de coordenadas EPSG. Permitirá que se execute operações ST_Transform() em geometrias.
9. UPDATE pg_database :marcar a nova base de dados postgistemplate como um template.

OBS: opcionalmente, se desejar um “help”(comentários) das descrições das funções incluídas no PostGIS, instalar o arquivo “postgis_comments.sql”. Os comentários poderão ser vistos simplesmente digitando `\dd [function_name]` a partir de uma janela de terminal psql. Proceder assim:

```
postgres$ psql -d postgistemplate -f
/usr/share/postgresql/8.4/contrib/postgis_comments.sql
```

O template já está pronto: fora criada o banco de dados “postgistemplate”, com várias funções disponíveis e com duas tabelas (“geometry_columns” e “spatial_ref_sys”). Agora podemos testar o funcionamento do banco de dados postgistemplate que acabamos de criar:

```
postgres$ psql -d postgistemplate -c "SELECT postgis_full_version();"
postgis_full_version
-----------------------------------------------------------------------
POSTGIS="1.5.1" GEOS="3.2.0-CAPI-1.6.0" PROJ="Rel. 4.7.1, 23 September 2009" LIBXML="2.7.7"
USE_STATS
(1 registro)
```

Como mostra a resposta ao comando, vemos o PostGIS em funcionamento com sua versão 1.5. Se a máquina estiver com o phpPgAdmin, acesse este "front-end" (se o phpPgAdmin estiver instalado na máquina local clique aqui para fazer o acesso (http://localhost/phppgadmin/)) e observe o banco de dados criado, as tabelas e as funções espaciais já disponíveis.

Verificar o que fora criado

```
psql=# \l+
                                                                  Lista do
+-----------------+----------+-------------+-------------+-----------
|      Nome       |   Dono   | Codificação |  Collation  |    Ctype
+-----------------+----------+-------------+-------------+-----------
| postgistemplate | gis      | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-
| postgres        | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-
| template0       | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-
|
| template1       | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-
|
+-----------------+----------+-------------+-------------+-----------
```

3.5) Atribuir permissões as tabelas
Atribuir permissões para as tabelas do banco de dados "postgistemplate" ("geometry_columns" e "spatial_ref_sys"), que terão como proprietário o usuário "gis"). Sair da conexão anterior (digite \q), e se conectar ao banco de dados "postgistemplate" ainda como usuário postgres e: atribuir permissões, criar um esquema para os dados gis (não se deve criar os dados gis em esquema "public") e, por último, desfazer a conexão com o banco de dados. Para isto, realizar os seguintes comandos:

```
postgres$ psql -d postgistemplate
psql=# ALTER TABLE geometry_columns OWNER TO gis;
psql=# ALTER TABLE spatial_ref_sys OWNER TO gis;
psql=# CREATE SCHEMA gis_schema AUTHORIZATION gis;
psql=# \q
```

4) A tabela SPATIAL_REF_SYS Table e sistemas de referencia espacial
Existem duas tabelas de meta-dados do OpenGIS: SPATIAL_REF_SYS e GEOMETRY_COLUMNS. Estas tabelas de sistema do PostGIS são criadas automaticamente quando da criação de uma base de dados. A tabela SPATIAL_REF_SYS carrega os identificadores numéricos e descrições textuais de sistemas de coordenadas utilizadas no banco de dados espacial.

A tabela spatial_ref_sys é uma inclusão do PostGIS e compatível com a tabela de base de dados da OGC que lista mais de 3000 sistemas de referência espacial conhecidos e possibilita realizar diferentes combinações de projeções e "datums", necessárias para mostrar com precisão dados geográficos de todas as áreas do planeta. Deve-se ter em mente que a maioria dos sistemas de referência espaciais são regionais e não têm significado quando usado fora dos limites que foram destinados.

Uma excelente fonte de informações para encontrar sistemas de referência espacial não definidos de forma centralizada e pública pode ser encontrado em http://spatialreference.org/ (http://spatialreference.org/)

Quando instalado o PostGIS, a sua documentação disponível localmente em `file:///usr/share/doc/postgis/postgis/postgis.html#id313360` traz mais detalhes sobre estas tabelas.

5) Criar base de dados

Após os procedimento anteriores, podem ser criados os bancos de dados necessários. Como exemplo neste post, será criado o banco de dados "gisdb", cujo proprietário será o usuário "gis", utilizando o template "postgistemplate":

- Do shell:
  - `postgres$ createdb gisdb -T postgistemplate -E utf-8 -O gis "postgis database"`
- Do psql:
  - ```
    postgres$ psql
    psql=# CREATE DATABASE gisdb with TEMPLATE = postgistemplate OWNER = gis ENCODING = 'utf8';
    psql=# COMMENT ON DATABASE gisdb IS 'postgis database';
    psql=# \q
    ```

Para verificar:

```
psql=# \l+
                                                                  Lista dos
+-----------------+----------+-------------+-------------+-----------
|      Nome       |   Dono   | Codificação |  Collation  |    Ctype
+-----------------+----------+-------------+-------------+-----------
| gisdb           | gis      | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-8
| postgistemplate | gis      | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-8
| postgres        | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-8
| template0       | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-8
|
| template1       | postgres | UTF8        | pt_BR.UTF-8 | pt_BR.UTF-8
|
+-----------------+----------+-------------+-------------+-----------
```

5.1) Carregar base de dados

- Gerar o arquivo .sql
  O carregador de dados shp2pgsql converte arquivos ESRI Shape em declarações adequadas SQL para inserção posterior em um banco de dados PostGIS/PostgreSQL tanto no formato de geometria como no de geografia. A título de exemplo, seria realizado da seguinte forma:
  `shp2pgsql -s 4674 -W LATIN1 12UFE250GC_SIR.dbf ufe12250 > ufe12250.sql`
  O que resulta o comando acima? Um arquivo sql com comandos para criar uma tabela entitulada "ufe12250" e tendo também uma coluna com o mesmo nome.
  OBS: é importante que o nome da tabela que irá ser gerada posteriormente na base de dados não inicie com um dígido, e sim por um caracter alfabético. A razão disto é que algumas transações XML exigem que os nomes dos layers/tabelas estejam iniciados por este tipo de caracter.
```

- Carregar os dados no banco de dados "gisdb"
  `$ psql -d gisdb -U gis -f ufe12250.sql`
  OBS: para evitar erro de conexão neste comando (conforme melhor descrito no post "Tutorial PostgreSQL:instalar em ambiente Linux – Debian (http://wp.me/pCWn2-eJ)"), certificar-se que no arquivo de configuração do PostgreSQL /etc/postgresql/8.4/main/pg_hba.conf possui uma entrada semelhante a:

```
local   all         all                   md5
```

6) Um pouco sobre "Sistema de Referência Espacial" (SRS)
Sistema de referência espacial (SRS) é um mecanismo para situar mensurações sobre um corpo geométrico, tal como a terra; estabelece um ponto de origem, orientação de eixos de referência, e significado geométrico das mensurações, assim como unidades de medida. De outra forma, um SRS é um conjunto de parâmetros usado para representar uma geometria.

É comum utilizar apenas um número para nos referir a uma referência espacial. Este número, é o SRID. Veja estes dois exemplos: EPSG 4326 e EPSG 4674. Nestes casos, temos:

- EPSG 4326 se refere ao GCS WGS84;
- EPSG 4674 se refere ao GCS SIRGAS-2000.

Obs:
i) GCS – geographic coordinate system;
ii) EPSG – é uma codificação definida pelo "European Petroleum Survey Group" que associa uma codificação numérica a um sistema de coordenadas cartográficas.

O CGS WGS84 é o sistema geodésico desenvolvido pelo Departamento da Defesa dos Estados Unidos da América. É o sistema de referência atualmente utilizado pelo GPS. A origem das coordenadas deste sistema geodésico é o centro da Terra, obtendo-se um erro é inferior a 2cm.

O GCS SIRGAS-2000, Sistema de Referencia Geocêntrico para as Américas, é o sistema de referenciamento espacial padrão do Brasil, definido pelo IBGE. Foi oficialmente adotado como Referencial Geodésico Brasileiro em 2005, através da Resolução do IBGE N°1/2005. Na prática o SIRGAS-2000 utiliza o mesmo elipsóde de referência do WGS84 – GRS 1980. O mesmo ocorre com o posicionamento do elipsóide, com seus centros situando-se bem próximos um do outro. Isto significa que as medidas dados pelo SIRGAS-2000 são quase iguais aos dados pelo WGS84, onde uma diferença entre as medidas pode ser menor que o erro aproximado de um instrumento GPS pessoal, pois esta diferença está na ordem de centímetros. Desta forma, para fins práticos, ou seja, para todos os usuários que não precisam de qualidade superior ao centímetro, é indiferente usar WGS84 ou SIRGAS-2000.

Sobre este tópico, existem boas referências na internet, entre as quais:
– FAQ IBGE (http://www.ibge.gov.br/home/geociencias/geodesia/pmrg/faq.shtm)
– Sistema de Referenciamento Espacial (http://seiti.eti.br/blog/2010/sistema-de-referenciamento-espacial)

7) A malha digital de municípios publicado pelo IBGE
O IBGE publica a malha digital de municípios a cada censo realizado. No momento deste post, o último censo demográfico fora realizado em 2010 e a malha digital referente pode ser baixada de `ftp://geoftp.ibge.gov.br/malhas_digitais/municipio_2010/` . As bases cartográficas disponibilizadas são compatíveis com a escala 1:250.000, sem supressão de pontos.

Este produto cartográfico do IBGE apresenta as seguintes unidades territoriais: municípios, microrregiões, mesorregiões e unidades da Federação. As bases cartográficas utilizam as seguintes referência geodésica e cartográfica:
– Sistema Geográfico – Sistema de Coordenadas Lat / Long – não projetado

– Sistema Geodésico – SIRGAS2000

Os arquivos shapefiles disponibilizados pelo IBGE estão em 3D, e para facilitar sua utilização pelo serviço WFS com o Geoserver (a partir de clientes como o OpenLayers, QGIS e outros) sugerimos fazer ajustes nos arquivos shapefiles. É que o Geoserver parece ainda se mostrar imaturo para tratar o 3D, pelo menos no momento da publicação deste post. No serviço WMS a terceira dimensão é removida, porém o mesmo código de remoção parece que ainda não está disponível para o caso do WFS.

O ajuste sugerido é transformar as fontes shapefiles de 3D para 2D antes de subir as informações cartográficas para o banco de dados. Um método fácil para esta transformação 3D/2D é utilizando a biblioteca GDAL (Geospatial Data Abstraction Library). GDAL é uma biblioteca de tradução para formatos de dados raster geospatial. Está incluída nesta biblioteca o código da biblioteca OGR que possui funcionalidades semelhantes às da GDAL, só que destinada para se trabalhar com dados vetorias (Simple Features). É esta biblioteca OGR que iremos utilizar.

```
/* instalar a biblioteca GDAL */
# apt-get install gdal-bin
/* se a bilbioteca OGR fora instalada com sucesso, o comando a seguir
$ ogrinfo
Usage: ogrinfo [--help-general] [-ro] [-q] [-where restricted_where]
               [-spat xmin ymin xmax ymax] [-fid fid]
               [-sql statement] [-al] [-so] [-fields={YES/NO}]
               [-geom={YES/NO/SUMMARY}][--formats]
               datasource_name [layer [layer ...]]
```

Para alterar o arquivo shapefile de 3D para 2D e já disponibilizar as informações numa tabela do PostgreSQL, poderia ser executado o seguinte comando:

```
$ogr2ogr -update -append -f PostgreSQL PG:"dbname=nome_BD
user='usuario_BD' password='senha_BD'" 12UFE250GC_SIR.shp -s_srs
EPSG:4674 -nln ufe12250b -nlt MULTIPOLYGON
```

Onde:
-f nome do formato do arquivo de saída. No caso, PostgreSQL.
-update: abrir a fonte de dados exitente de saída no modo “update”, em vez de tentar criar uma nova.
-append: anexa as informações para o layer existente em vez de criar um novo
-overwrite: apaga o layer de saída e recria um novo (embora que não fora utilizado neste exemplo)
-s_srs: substitui a fonte SRS na saída
-nln: nome do novo layer, ou seja, um nome para a tabela que será criada.
-nlt: define o tipo da geometria para o layer que será criado. Pode ser: NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON or MULTILINESTRING.

Nos testes que realizei, fiz um procedimento mais “controlado” em 3 etapas, conforme abaixo. E funcionou tudo perfeitamente:

```
$ ogr2ogr -t_srs EPSG:4674 -nlt MULTIPOLYGON ma/arq_novo4674.shp
ma/21UFE250GC_SIR.shp
$ shp2pgsql -s 4674 -W LATIN1 ma/arq_novo4674.shp ufemaranhao >
ma/ufemaranhao.sql
$ psql -d nome_BD -U usuario_BD -f ma/ufemaranhao.sql
```

Neste exemplo acima, tem-se:
– arq_novo4674: arquivos shapefiles que serão gerados (.dbf, .prj, .shp, .shx) pelo comando ogr2ogr, já na forma 2D;

– ma/21UFE250GC_SIR.shp: arquivo de entrada, referente ao estado do Maranhão, que tomamos a partir do original fornecido pelo IBGE. Esse arquivo está na pasta "ma" logo abaixo do diretório corrente de trabalho do usuário;
– ufemaranhao: nome da tabela que será gerada posteriormente no banco de dados PostgreSQL, qundo da execução do psql a seguir;
– ufemaranhao.sql: nome do arquivo com comandos SQL que será gerado pela execução do shp2pgsql.

Obs: sobre como o PostgreSQL trata os conjuntos de caracteres (ISO-8859 e UTF-8, por exemplo), ver o manual do PostgreSQL na sessão Character Set Support (http://www.postgresql.org/docs/8.4/static/multibyte.html).

7.1 Tamanho dos arquivos
Sempre se deve ter cuidado com o tamanho dos arquivos. Quanto maior a escala, maior o arquivo. Veja o resultado comparativo para o estado de Pernambuco entre as malhas digitais fornecidas pelo IBGE da escala 1:250k e 1:2500k, conforme abaixo:

IBGE-2010 (1:250K)	**IBGE-2007 (1:2500K)**
shapefile 26MUE250GC_SIR.shp `1608kB`	shapefile 26mu2500gsr.shp `261kB`
download para serviço WFS de `1100kB`	download para serviço WFS de `278kB`

8) O arquivo de Password .pgpass
Uma forma bastante prática de acessar o banco de dados PostgreSQL sem ter a necessidade de se digitar todas as vezes o password é criando o arquivo de autenticação (password) `.pgpass`, conforme orienta o manual do PostgreSQL na sessão The Password File (http://www.postgresql.org/docs/8.4/static/libpq-pgpass.html). Isso pode ser um facilitador quando se utiliza recorrentemente comandos psql.

Este arquivo de password deve ser colocado no diretório home do usuário. Desta forma, o usuário PostgreSQL pode conectar ao banco sem informar senha, pois a mesma já estará informada neste arquivo. O formato deste arquivo é o seguinte:

```
hostname:port:database:username:password
```

Veja este exemplo: localhost:5432:*:jose:senha_jose
Onde deve ser observado:
– jose deve ser um usuário do banco PostgreSQL e não do S.O;
– no caso do usuário root, o arquivo deve estar em em /root/.pgpass
– este arquivo deve ter obrigatoriamente a permissão 0600 (comando "`chmod 0600 .pgpass`"). Se a permissão for menos restritiva que 0600, o arquivo será ignorado.

9) Bibliotecas complementares
As seguintes bibliotecas estão disponíveis, mas não são obrigatórias para um uso inicial do PostGIS:

```
# apt-get install proj libgeos-c1 libgeos-dev
```

Proj4 é uma biblioteca responsável por conversões de projeções.
GEOS (Geometry Engine – Open Source) és uma porta C++ da Java Topology Suite (JTS). Ela objetiva alcançar uma completa funcionalidade do JTS no C++. Ou seja, a total compatibilidade do PostGIS com a SFS.

Referências:
1- Tutorial PostgreSQL (instalar em ambiente Linux – Debian) (http://wp.me/pCWn2-eJ)
2- Manual: installation PostGIS (http://postgis.refractions.net/documentation/manual-1.3/ch02.html)
3- GIS Tutorials (http://www.gistutor.com/)
4- Installing PostGIS on debian GNU/Linux (http://blog.edseek.com/archives/2004/03/31/installing-postgis-on-debian-gnulinux/)
5- Installing PostGIS on Ubuntu (http://www.paolocorti.net/2008/01/30/installing-postgis-on-ubuntu/)
6- Installing postgis and mapserver in Debian (http://www.duif.net/postgis/)

7- Debian Postgis (http://www.jtheuer.de/pub/debian_postgis_postresql-8.0)
8- Importando/Exportando dados vetoriais ao PostGIS (http://geoind.wordpress.com/2013/06/17/vetoriais_postgis/)
—-
9- OGR SQL (http://www.gdal.org/ogr/ogr_sql.html)
10- Malha geométrica dos municípios brasileiros (http://dados.gov.br/dataset/malha-geometrica-dos-municipios-brasileiros)
11- Reprojecting features using OGR (http://beginningspatial.com/reprojecting_features_using_ogr)
12- OGR to reproject, modify Shapefiles (https://github.com/nvkelso/geo-how-to/wiki/OGR-to-reproject,-modify-Shapefiles)

comentários

Bruno de Freitas Romanhooli disse:
26/07/2012 às 12:15
Olá, estou com problemas com um arquivo.
O postgis.sql
Ele não se encontra no diretório indicado. Tentei dar um locate, mas ele encontrou outros arquivos e não expecificamente esse.. os encontrados foram:
postgis.sql.in.c
uninstall_postgis.sql.in.c
postgis.sql.in.c
uninstall_postgis.sql.in.c

Será que fiz algo errado na instalação do postgis??

Agradeço desde já a atenção.

Responder

felipe disse:
07/07/2012 às 16:23
Excelente post!
Muito útil para quem está comecando no mundo Gis!
Abracos

Responder

Blog no WordPress.com.