PostGIS 2.2.0dev 开发手册

581 / 686

**Chapter 12**

**PostGIS 扩充**

本章文档介绍的东西可以在PostGIS源码tar包或者源码repository地址中找到。这些特性不会打包在PostGIS二进制版本中，但通常都是基于plpgsql编写的或者标准的可运行的shell脚本编写。

**12.1**

**Tiger(地理编码和参考系统的拓扑集成) Geocoder**

对于PostGIS来说有许多开源的地理编码工具，与tiger地理编码器不同的是还有多国家地理编码的支持。

· Nominatim它使用OpenStreetMap 分类格式数据。它需要osm2pgsql来加载工具，并且需要数据库环境有PostgreSQL 8.4+

和 PostGIS 1.5+ 才能使用。它被打包成一个web服务接口并且被设计成一个服务调用。和tiger地理编码器类似的是，它有一个地理转码和地理逆转码组件。从它的官方文档来看，关于它是否有一个像tiger 地理编码这样的纯SQL接口问题，或者在web接口中是否有一个比较好的逻辑处理，有些含糊不清。

·GIS Graphy同样利用PostGIS系统环境，并且和Nominatim一样使用OpenStreetMap (OSM) 数据，并且使用加载工具来加载OSM数据，和Nominatim类似的是支持的地理编码不止是US的。和Nominatim还类似的是，它以一个web服务的方式运行，并且依赖环境Java 1.5, Servlet应用, Solr. GisGraphy是跨平台的，除了其他灵巧的特性外，它也有一个逆地理转码组件。

**12.1.1**

**Drop\_Indexes\_Generate\_Script**

Drop\_Indexes\_Generate\_Script —这个函数会生成一个脚本，用于drop掉所有tiger schema和用户指定的schema下的非primary key和非唯一性索引之外的索引。如果没有指定schema，那么默认的schema是tiger\_data。

**用法**

text Drop\_Indexes\_Generate\_Script(text param\_schema=tiger\_data);

**描述**

这个函数会生成一个脚本，用于drop掉所有tiger schema和用户指定的schema下的非primary key和非唯一性索引之外的索引。如果没有指定schema，那么默认的schema是tiger\_data。

这个对于最小化索引膨胀（索引膨胀会误导查询优化器）或者减少不必要的空间很有用。这个函数可以和函数Install\_Missing\_Indexes一起使用，用于只添加地理编码所需的索引。

可用版本: 2.0.0。

PostGIS 2.2.0dev 开发手册

582 / 686

**样例**

SELECT drop\_indexes\_generate\_script() As actionsql;

actionsql

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

DROP INDEX tiger.idx\_tiger\_countysub\_lookup\_lower\_name;

DROP INDEX tiger.idx\_tiger\_edges\_countyfp;

DROP INDEX tiger.idx\_tiger\_faces\_countyfp;

DROP INDEX tiger.tiger\_place\_the\_geom\_gist;

DROP INDEX tiger.tiger\_edges\_the\_geom\_gist;

DROP INDEX tiger.tiger\_state\_the\_geom\_gist;

DROP INDEX tiger.idx\_tiger\_addr\_least\_address;

DROP INDEX tiger.idx\_tiger\_addr\_tlid;

DROP INDEX tiger.idx\_tiger\_addr\_zip;

DROP INDEX tiger.idx\_tiger\_county\_countyfp;

DROP INDEX tiger.idx\_tiger\_county\_lookup\_lower\_name;

DROP INDEX tiger.idx\_tiger\_county\_lookup\_snd\_name;

DROP INDEX tiger.idx\_tiger\_county\_lower\_name;

DROP INDEX tiger.idx\_tiger\_county\_snd\_name;

DROP INDEX tiger.idx\_tiger\_county\_the\_geom\_gist;

DROP INDEX tiger.idx\_tiger\_countysub\_lookup\_snd\_name;

DROP INDEX tiger.idx\_tiger\_cousub\_countyfp;

DROP INDEX tiger.idx\_tiger\_cousub\_cousubfp;

DROP INDEX tiger.idx\_tiger\_cousub\_lower\_name;

DROP INDEX tiger.idx\_tiger\_cousub\_snd\_name;

DROP INDEX tiger.idx\_tiger\_cousub\_the\_geom\_gist;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_addr\_least\_address;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_addr\_tlid;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_addr\_zip;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_county\_countyfp;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_county\_lookup\_lower\_name;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_county\_lookup\_snd\_name;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_county\_lower\_name;

DROP INDEX tiger\_data.idx\_tiger\_data\_ma\_county\_snd\_name;

:

:

**请参考**

Install\_Missing\_Indexes, Missing\_Indexes\_Generate\_Script

**12.1.2**

**Drop\_Nation\_Tables\_Generate\_Script**

Drop\_Nation\_Tables\_Generate\_Script —生成一个SQL脚本，这个脚本用于drop掉指定schema的所有表。生成的SQL脚本drop的表从county\_all开始drop。

**用法**

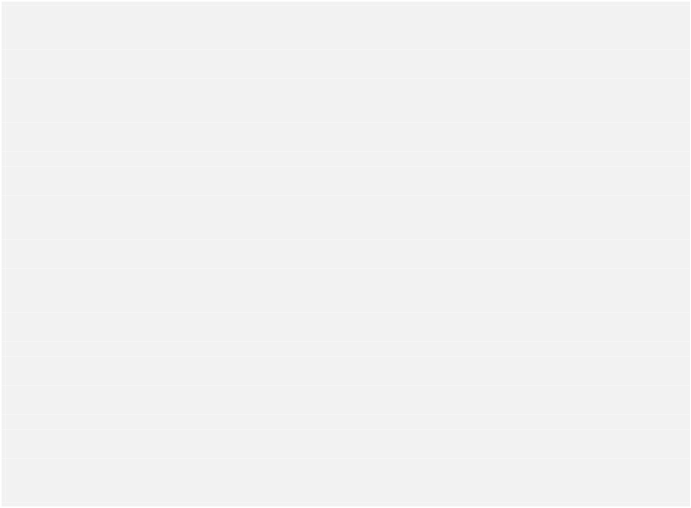
text Drop\_Nation\_Tables\_Generate\_Script(text param\_schema=tiger\_data);

**描述**

生成一个SQL脚本，这个脚本用于drop掉指定schema的所有表。生成的SQL脚本drop的表从county\_all开始drop。如果你想把

tiger\_2010数据升级到tiger\_2011 数据的话，这个很有用。

可用版本: 2.1.0。



PostGIS 2.2.0dev 开发手册

583 / 686

**样例**

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

DROP TABLE tiger\_data.county\_all;

DROP TABLE tiger\_data.county\_all\_lookup;

DROP TABLE tiger\_data.state\_all;

DROP TABLE tiger\_data.ma\_county;

DROP TABLE tiger\_data.ma\_state;

**请参考**

Loader\_Generate\_Nation\_Script

**12.1.3**

**Drop\_State\_Tables\_Generate\_Script**

Drop\_State\_Tables\_Generate\_Script —生成一个SQL脚本，这个脚本可以drop掉一个指定schema下的所有以某个州缩写字母作为前缀的所有表。如果没有指定schema的话，默认的schema是tiger\_data

**用法**

text Drop\_State\_Tables\_Generate\_Script(text param\_state, text param\_schema=tiger\_data);

**描述**

生成一个SQL脚本，这个脚本可以drop掉一个指定schema下的所有以某个州缩写字母作为前缀的所有表。如果没有指定schema的话，默认的schema是tiger\_data。这个函数是很有用的，比如在你加载数据出错后，需要再次加载数据前，需要把以某个州开头的表全部drop掉，那么就可以使用这个函数。

可用版本: 2.0.0。

**样例**

SELECT drop\_state\_tables\_generate\_script(’PA’);

DROP TABLE tiger\_data.pa\_addr;

DROP TABLE tiger\_data.pa\_county;

DROP TABLE tiger\_data.pa\_county\_lookup;

DROP TABLE tiger\_data.pa\_cousub;

DROP TABLE tiger\_data.pa\_edges;

DROP TABLE tiger\_data.pa\_faces;

DROP TABLE tiger\_data.pa\_featnames;

DROP TABLE tiger\_data.pa\_place;

DROP TABLE tiger\_data.pa\_state;

DROP TABLE tiger\_data.pa\_zip\_lookup\_base;

DROP TABLE tiger\_data.pa\_zip\_state;

DROP TABLE tiger\_data.pa\_zip\_state\_loc;

**请参考**

Loader\_Generate\_Script

PostGIS 2.2.0dev 开发手册

584 / 686

**12.1.4**

**Geocode**

Geocode —获取一个字符串地址（或者其他标准化的地址），然后输出一个可能的地址集合，这些地址都包含一个point几何对象（以NAD83 参考系下的经纬度形式）和一个标准化地址以及一个地址匹配程度的评估值rating，rating值越低表示这个输出地址越匹配输入的地址。输出结果按照rating升序排列（即最匹配的放在前面）。这个函数可以设置一个最大的输出结果记录数，默认是10条记录和限制区域参数restrict\_region（默认为NULL）

**用法**

setof record geocode(varchar address, integer max\_results=10, geometry restrict\_region=NULL, norm\_addy OUT addy, geom-

etry OUT geomout, integer OUT rating);

setof record geocode(norm\_addy in\_addy, integer max\_results=10, geometry restrict\_region=NULL, norm\_addy OUT addy,

geometry OUT geomout, integer OUT rating);

**描述**

获取一个字符串地址（或者其他标准化的地址），然后输出一个可能的地址集合，这些地址都包含一个point几何对象（以NAD83 参考系下的经纬度形式）和一个标准化地址（addy）以及一个地址匹配程度的评估值rating，rating值越低表示这个输出地址越匹配输入的地址。输出结果按照rating升序排列（即最匹配的放在前面）。

这个函数会使用Tiger 地理数据 (edges,faces,addr), PostgreSQL的 fuzzy字符串匹配(soundex,levenshtein) 和PostGIS的线性插值函数来沿着Tiger的边进行地址插值匹配。输出结果字段rating值越大，输出地址和输入的地址越不匹配。被解码的点默认最多从街道地址的中心线偏离10米。

版本提升: 2.0.0版本开始支持Tiger 2010 版本结构数据，并且修改了部分处理逻辑来提高速度，解码精确度。新的参数max\_results 对于只想返回最佳结果的使用场景很有用。

**样例: Basic**

The below examples timings are on a 3.0 GHZ single processor Windows 7 machine with 2GB ram running PostgreSQL

9.1rc1/PostGIS 2.0 loaded with all of MA,MN,CA, RI state Tiger data loaded.

Exact matches are faster to compute (61ms)

SELECT g.rating, ST\_X(g.geomout) As lon, ST\_Y(g.geomout) As lat,

(addy).address As stno, (addy).streetname As street,

(addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,(addy) ←

.zip

FROM geocode(’75 State Street, Boston MA 02109’) As g;

rating |

lon

|

lat

| stno | street | styp |

city

| st |

zip

--------+-------------------+------------------+------+--------+------+--------+----+------- ←

0 | -71.0556722990239 | 42.3589914927049 |

75 | State

| St

| Boston | MA | 02109

Even if zip is not passed in the geocoder can guess (took about 122-150 ms)

SELECT g.rating, ST\_AsText(ST\_SnapToGrid(g.geomout,0.00001)) As wktlonlat,

(addy).address As stno, (addy).streetname As street,

(addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,(addy) ←

.zip

FROM geocode(’226 Hanover Street, Boston, MA’,1) As g;

rating |

wktlonlat

| stno | street

| styp |

city

| st |

zip

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

1 | POINT(-71.05528 42.36316) | 226 | Hanover | St

| Boston | MA | 02113

Can handle misspellings and provides more than one possible solution with ratings and takes longer (500ms).

PostGIS 2.2.0dev 开发手册

585 / 686

SELECT g.rating, ST\_AsText(ST\_SnapToGrid(g.geomout,0.00001)) As wktlonlat,

(addy).address As stno, (addy).streetname As street,

(addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,(addy) ←

.zip

FROM geocode(’31 - 37 Stewart Street, Boston, MA 02116’) As g;

rating |

wktlonlat

| stno | street | styp |

city

| st |

zip

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

70 | POINT(-71.06459 42.35113) | 31 | Stuart | St

| Boston | MA | 02116

Using to do a batch geocode of addresses. Easiest is to set max\_results=1. Only process those not yet geocoded (have no

rating).

CREATE TABLE addresses\_to\_geocode(addid serial PRIMARY KEY, address text,

lon numeric, lat numeric, new\_address text, rating integer);

INSERT INTO addresses\_to\_geocode(address)

VALUES (’529 Main Street, Boston MA, 02129’),

(’77 Massachusetts Avenue, Cambridge, MA 02139’),

(’25 Wizard of Oz, Walaford, KS 99912323’),

(’26 Capen Street, Medford, MA’),

(’124 Mount Auburn St, Cambridge, Massachusetts 02138’),

(’950 Main Street, Worcester, MA 01610’);

-- only update the first 3 addresses (323-704 ms -

there are caching and shared memory

←

effects so first geocode you do is always slower) --

-- for large numbers of addresses you don’t want to update all at once

-- since the whole geocode must commit at once

-- For this example we rejoin with LEFT JOIN

-- and set to rating to -1 rating if no match

-- to ensure we don’t regeocode a bad address

UPDATE addresses\_to\_geocode

SET

(rating, new\_address, lon, lat)

= ( COALESCE((g.geo).rating,-1), pprint\_addy((g.geo).addy),

ST\_X((g.geo).geomout)::numeric(8,5), ST\_Y((g.geo).geomout)::numeric(8,5) )

FROM (SELECT addid

FROM addresses\_to\_geocode

WHERE rating IS NULL ORDER BY addid LIMIT 3) As a

LEFT JOIN (SELECT addid, (geocode(address,1)) As geo

FROM addresses\_to\_geocode As ag

WHERE ag.rating IS NULL ORDER BY addid LIMIT 3) As g ON a.addid = g.addid

WHERE a.addid = addresses\_to\_geocode.addid;

result

-----

Query returned successfully: 3 rows affected, 480 ms execution time.

SELECT \* FROM addresses\_to\_geocode WHERE rating is not null;

addid |

address

|

lon

|

lat

|

←

new\_address

| rating

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

1 | 529 Main Street, Boston MA, 02129 | -71.07181 | 42.38359 | 529 Main St, ←

Boston, MA 02129 | 0

2 | 77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428 | 42.35988 | 77

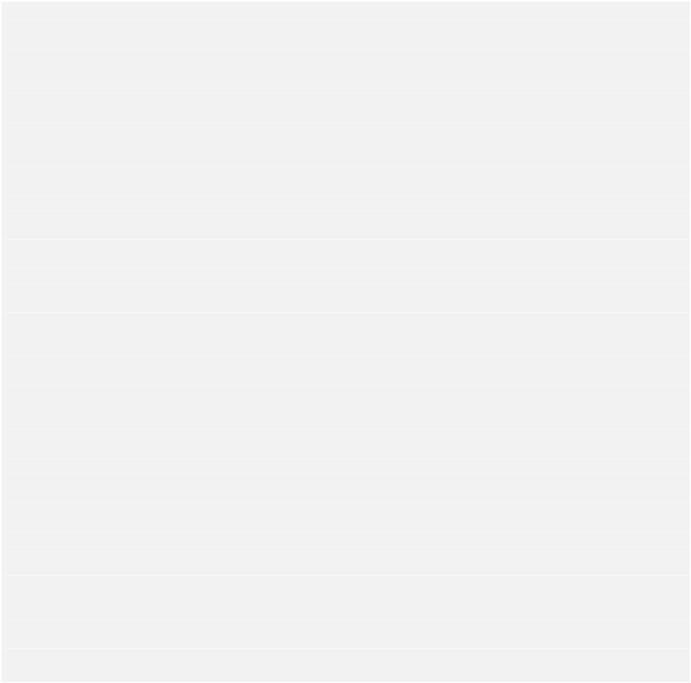
Massachusetts Ave, Cambridge, MA 02139 | 0

←

3 | 25 Wizard of Oz, Walaford, KS 99912323 | | |

| -1

←



PostGIS 2.2.0dev 开发手册

586 / 686

**样例: Using Geometry ﬁlter**

SELECT g.rating, ST\_AsText(ST\_SnapToGrid(g.geomout,0.00001)) As wktlonlat,

(addy).address As stno, (addy).streetname As street,

(addy).streettypeabbrev As styp,

(addy).location As city, (addy).stateabbrev As st,(addy).zip

FROM geocode(’100 Federal Street, MA’,

3,

(SELECT ST\_Union(the\_geom)

FROM place WHERE statefp = ’25’ AND name = ’Lynn’)::geometry

) As g;

rating |

wktlonlat

| stno | street

| styp | city | st |

zip

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

8 | POINT(-70.96796 42.4659) | 100 | Federal | St

Total query runtime: 245 ms.

**请参考**

Normalize\_Address, Pprint\_Addy, ST\_AsText, ST\_SnapToGrid, ST\_X, ST\_Y

| Lynn | MA | 01905

**12.1.5**

**Geocode\_Intersection**

Geocode\_Intersection — 输入参数是两条相交的街道以及参数state, city, zip，然后输出两条街道第一个交叉点的可能地址。这些地址都包含一个point几何对象（以NAD83 参考系下的经纬度形式）和一个标准化地址以及一个地址匹配程度的评估值rating，rating值越低表示这个输出地址越匹配输入的地址。输出结果按照rating升序排列（即最匹配的放在前面）。这个函数可以设置一个最大的输出结果记录数，默认是10条记录。

**用法**

setof record geocode\_intersection(text roadway1, text roadway2, text in\_state, text in\_city, text in\_zip, integer max\_results=10,

norm\_addy OUT addy, geometry OUT geomout, integer OUT rating);

**描述**

输入参数是两条相交的街道以及参数state, city, zip，然后输出两条街道第一个交叉点的可能地址。这些地址都包含一个point几何对象（以NAD83 参考系下的经纬度形式）和一个标准化地址以及一个地址匹配程度的评估值rating，rating值越低表示这个输出地址越匹配输入的地址。输出结果按照rating升序排列（即最匹配的放在前面）。这个函数可以设置一个最大的输出结果记录数，默认是10条记录。这个函数会使用Tiger 地理数据 (edges,faces,addr), PostgreSQL的 fuzzy字符串匹配(soundex,levenshtein) 和PostGIS的线性插值函数来沿着Tiger的边进行地址插值匹配。

可用版本: 2.0.0。

**样例: Basic**

The below examples timings are on a 3.0 GHZ single processor Windows 7 machine with 2GB ram running PostgreSQL 9.0/Post-

GIS 1.5 loaded with all of MA state Tiger data loaded. Currently a bit slow (3000 ms)

Testing on Windows 2003 64-bit 8GB on PostGIS 2.0 PostgreSQL 64-bit Tiger 2011 data loaded -- (41ms)

SELECT pprint\_addy(addy), st\_astext(geomout),rating

FROM geocode\_intersection( ’Haverford St’,’Germania St’, ’MA’, ’Boston’, ’02130’,1);

pprint\_addy

|

st\_astext

| rating

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

98 Haverford St, Boston, MA 02130 | POINT(-71.101375 42.31376) | 0

PostGIS 2.2.0dev 开发手册

587 / 686

Even if zip is not passed in the geocoder can guess (took about 3500 ms on the windows 7 box), on the windows 2003 64-bit 741

ms

SELECT pprint\_addy(addy), st\_astext(geomout),rating

FROM geocode\_intersection(’Weld’, ’School’, ’MA’, ’Boston’);

pprint\_addy

|

st\_astext

| rating

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

98 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) | 3

99 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) | 3

**请参考**

Geocode, Pprint\_Addy, ST\_AsText

**12.1.6**

**Get\_Geocode\_Setting**

Get\_Geocode\_Setting —返回表tiger.geocode\_settings某个setting\_name的setting 值。

**用法**

text Get\_Geocode\_Setting(text setting\_name);

**描述**

返回表tiger.geocode\_settings某个setting\_name的setting 值。这个设置允许你进行函数的debugging 切换。这样解码地理位置的函数会根据settings控制rating。当前的settings列表如下：

name

| setting |

unit

| category

|

←

short\_desc

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

debug\_geocode\_address

| false

| boolean | debug

| outputs debug information ←

in notice log such as queries when geocode\_addresss is called if true

debug\_geocode\_intersection

| false

| boolean | debug

| outputs debug information ←

in notice log such as queries when geocode\_intersection is called if true

debug\_normalize\_address

| false

| boolean | debug

| outputs debug information ←

in notice log such as queries

| | | |

and intermediate

←

expressions when normalize\_address is called if true

debug\_reverse\_geocode

| false

| boolean | debug

| if true, outputs debug

←

information in notice log such as queries

|

and intermediate

expressions when

reverse\_geocode

←

←

reverse\_geocode\_numbered\_roads | 0 | integer | rating

highways, 0 - no preference in name

| For state and county

←

| | | | , 1 - prefer the

←

numbered highway name, 2 - prefer local state/county

name

←

use\_pagc\_address\_parser

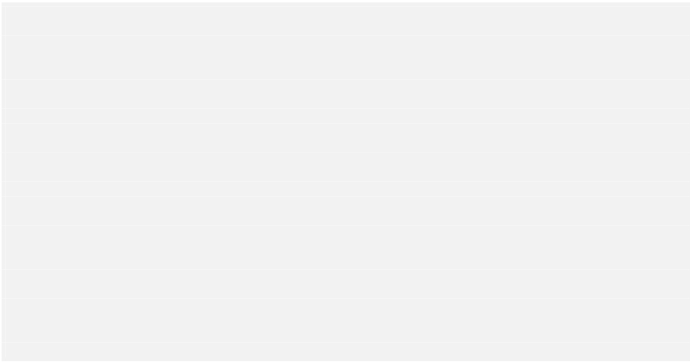
| false

| boolean | normalize | If set to true, will try

←

to use the pagc\_address normalizer instead of tiger built one

可用版本: 2.1.0。



PostGIS 2.2.0dev 开发手册

588 / 686

**Example return debugging setting**

SELECT get\_geocode\_setting(’debug\_geocode\_address) As result;

result

---------

false

**请参考**

Set\_Geocode\_Setting

**12.1.7**

**Get\_Tract**

Get\_Tract — 根据geometry几何对象所在的位置，从人口普查表中返回这个位置的人口普查数据。

**用法**

text get\_tract(geometry loc\_geom, text output\_ﬁeld=name);

**描述**

根据geometry几何对象所在的位置，从人口普查表中返回这个位置的人口普查数据。如果没有指定参考系，默认使用NAD 83经纬度作为参考系。

可用版本: 2.0.0。

**样例: Basic**

SELECT get\_tract(ST\_Point(-71.101375, 42.31376) ) As tract\_name;

tract\_name

---------

1203.01

--this one returns the tiger geoid

SELECT get\_tract(ST\_Point(-71.101375, 42.31376), ’tract\_id’ ) As tract\_id;

tract\_id

---------

25025120301

**请参考**

Geocode>

**12.1.8**

**Install\_Missing\_Indexes**

Install\_Missing\_Indexes — 找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了，然后添加这些缺失的索引。

**用法**

boolean Install\_Missing\_Indexes();

PostGIS 2.2.0dev 开发手册

589 / 686

**描述**

在schema 为tiger和tiger\_data中找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了。然后输出定义这些索引的SQL DDL语句，最后执行这些脚本来添加这些缺失的索引。这个函数是很有用的函数，它添加让查询更快的新的索引，而这些索引可能在加载数据过程中丢失了。这个函数可以和函数Missing\_Indexes\_Generate\_Script联合使用，后者生成定义索引的SQL语句，并需要执行这个脚本。这个函数在升级脚本update\_geocode.sql中会有调用到。

可用版本: 2.0.0。

**样例**

SELECT install\_missing\_indexes();

install\_missing\_indexes

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

t

**请参考**

Loader\_Generate\_Script, Missing\_Indexes\_Generate\_Script

**12.1.9**

**Loader\_Generate\_Census\_Script**

Loader\_Generate\_Census\_Script — 生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据,bg和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。

**用法**

setof text loader\_generate\_census\_script(text[] param\_states, text os);

**描述**

生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据,bg和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。

这个函数在Linux平台上使用unzip（在Windows平台上默认使用7-zip）和wget来下载数据，然后使用4.4.2 节来加载数据。注意这个函数处理的数据最小是一整个州的数据。它只会处理staging和temp下的目录。它使用下面的控制表来控制处理进程和不同操作系统的shell语法变量。

1. loader\_variables：变量表，记录不同变量的表，比如census site, year, data 和 staging schemas

2. loader\_platform ：不同平台的资料信息以及不同平台的可执行命令的位置。表的记录只带有Windows和Linux的平台的信息，可以添加更多平台的信息。

3. loader\_lookuptables 每条记录定义了一种类型的表 (state, county),是否要处理记录中的数据以及怎样加载这种类型的数据，并且定义了每一种类型导入数据，存储数据，添加、移除列、索引和约束的步骤。每个表的名字都以相应的州为前缀，并且从schema名称为tiger的表中继承过来。比如表tiger\_data.ma\_faces继承于表tiger.faces。

可用版本: 2.0.0。

**注意**

Loader\_Generate\_Script函数生成的脚本包含了这个函数的逻辑，但是如果你在PostGIS 2.0 alpha5版本之前安装了tiger geocoder，你需要在那些你需要的州上运行这个函数，以便获取这些数据。



PostGIS 2.2.0dev 开发手册

590 / 686

**样例**

Generate script to load up data for select states in Windows shell script format.

SELECT loader\_generate\_census\_script(ARRAY[’MA’], ’windows’);

-- result --

set STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\25\_Massachusetts"

set TMPDIR=\gisdata\temp\

set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"

set WGETTOOL="C:\wget\wget.exe"

set PGBIN=C:\projects\pg\pg91win\bin\

set PGPORT=5432

set PGHOST=localhost

set PGUSER=postgres

set PGPASSWORD=yourpasswordhere

set PGDATABASE=tiger\_postgis20

set PSQL="%PGBIN%psql"

set SHP2PGSQL="%PGBIN%shp2pgsql"

cd \gisdata

%WGETTOOL% http://www2.census.gov/geo/pvs/tiger2010st/25\_Massachusetts/25/ --no-parent -- ←

relative --accept=\*bg10.zip,\*tract10.zip,\*tabblock10.zip --mirror --reject=html

del %TMPDIR%\\*.\* /Q

%PSQL% -c "DROP SCHEMA tiger\_staging CASCADE;"

%PSQL% -c "CREATE SCHEMA tiger\_staging;"

cd %STATEDIR%

for /r %%z in (\*.zip) do %UNZIPTOOL% e %%z

-o%TMPDIR%

cd %TMPDIR%

%PSQL% -c "CREATE TABLE tiger\_data.MA\_tract(CONSTRAINT pk\_MA\_tract PRIMARY KEY (tract\_id) ) ←

INHERITS(tiger.tract); "

%SHP2PGSQL% -c -s 4269 -g the\_geom

ma\_tract10 | %PSQL%

-W "latin1" tl\_2010\_25\_tract10.dbf tiger\_staging. ←

%PSQL% -c "ALTER TABLE tiger\_staging.MA\_tract10 RENAME geoid10 TO tract\_id;

SELECT

←

loader\_load\_staged\_data(lower(’MA\_tract10’), lower(’MA\_tract’)); "

%PSQL% -c "CREATE INDEX tiger\_data\_MA\_tract\_the\_geom\_gist ON tiger\_data.MA\_tract USING gist ←

(the\_geom);"

%PSQL% -c "VACUUM ANALYZE tiger\_data.MA\_tract;"

%PSQL% -c "ALTER TABLE tiger\_data.MA\_tract ADD CONSTRAINT chk\_statefp CHECK (statefp =

←

’25’);"

:

Generate sh script

STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/25\_Massachusetts"

TMPDIR="/gisdata/temp/"

UNZIPTOOL=unzip

WGETTOOL="/usr/bin/wget"

export PGBIN=/usr/pgsql-9.0/bin

export PGPORT=5432

export PGHOST=localhost

export PGUSER=postgres

export PGPASSWORD=yourpasswordhere

export PGDATABASE=geocoder

PSQL=${PGBIN}/psql

SHP2PGSQL=${PGBIN}/shp2pgsql

cd /gisdata

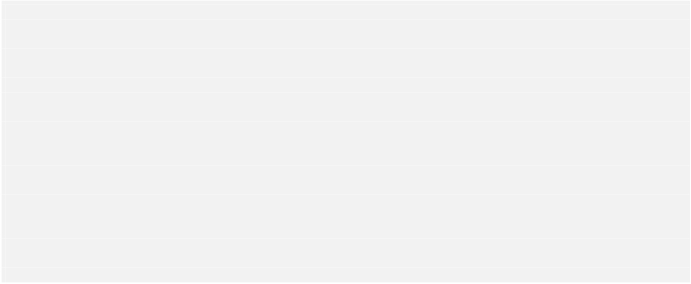
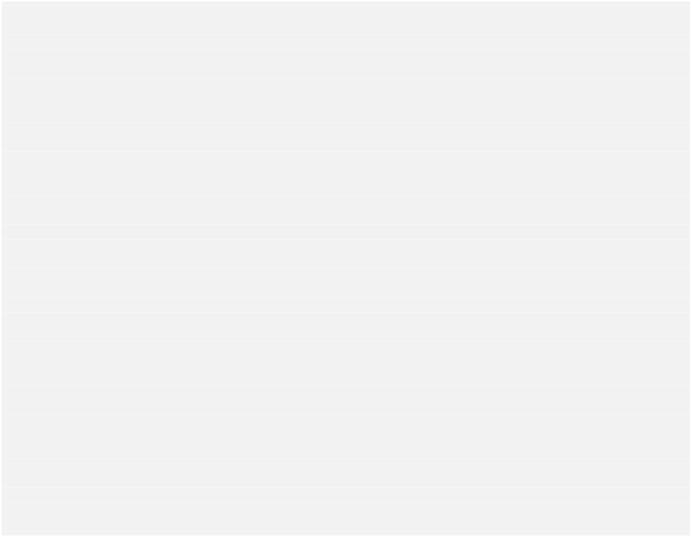
wget http://www2.census.gov/geo/pvs/tiger2010st/25\_Massachusetts/25/ --no-parent --relative ←

--accept=\*bg10.zip,\*tract10.zip,\*tabblock10.zip --mirror --reject=html

rm -f ${TMPDIR}/\*.\*

${PSQL} -c "DROP SCHEMA tiger\_staging CASCADE;"

${PSQL} -c "CREATE SCHEMA tiger\_staging;"



PostGIS 2.2.0dev 开发手册

591 / 686

cd $STATEDIR

for z in \*.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done

:

:

**请参考**

Loader\_Generate\_Script

**12.1.10**

**Loader\_Generate\_Script**

Loader\_Generate\_Script — 生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据,block groups （bg）和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。该函数的最新版本支持Tiger 2010结构化数据，并且会加载到census tract, block groups和 blocks tables表。

**用法**

setof text loader\_generate\_script(text[] param\_states, text os);

**描述**

生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据,bg和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。这个函数在Linux平台上使用unzip（在Windows平台上默认使用7-zip）和wget来下载数据，然后使用4.4.2 节来加载数据。注意这个函数处理的数据最小是一整个州的数据，但是你可以重新下载、覆盖已有的数据。它只会处理staging和temp下的目录。它使用下面的控制表来控制处理进程和不同操作系统的shell语法变量。

1. loader\_variables：变量表，记录不同变量的表，比如census site, year, data 和 staging schemas

2. loader\_platform ：不同平台的资料信息以及不同平台的可执行命令的位置。表的记录只带有Windows和Linux的平台的信息，可以添加更多平台的信息。

3. loader\_lookuptables 每条记录定义了一种类型的表 (state, county),是否要处理记录中的数据以及怎样加载这种类型的数据，并且定义了每一种类型导入数据，存储数据，添加、移除列、索引和约束的步骤。每个表的名字都以相应的州为前缀，并且从schema名称为tiger的表中继承过来。比如表tiger\_data.ma\_faces继承于表tiger.faces。

可用版本: 2.0.0。 支持Tiger 2010版本结构化数据，并且会加载如下表的数据： census tract (tract), block groups (bg)和 blocks (tabblocks)。

**样例**

Generate script to load up data for 2 states in Windows shell script format.

SELECT loader\_generate\_script(ARRAY[’MA’,’RI’], ’windows’) AS result;

-- result --

set STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\44\_Rhode\_Island"

set TMPDIR=\gisdata\temp\

set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"

set WGETTOOL="C:\wget\wget.exe"

set PGBIN=C:\Program Files\PostgreSQL\8.4\bin\

set PGPORT=5432

set PGHOST=localhost

PostGIS 2.2.0dev 开发手册

592 / 686

set

set

set

set

set

PGUSER=postgres

PGPASSWORD=yourpasswordhere

PGDATABASE=geocoder

PSQL="%PGBIN%psql"

SHP2PGSQL="%PGBIN%shp2pgsql"

%WGETTOOL% http://www2.census.gov/geo/pvs/tiger2010st/44\_Rhode\_Island/ --no-parent -- ←

relative --recursive --level=2 --accept=zip,txt --mirror --reject=html

:

:

Generate sh script

SELECT loader\_generate\_script(ARRAY[’MA’,’RI’], ’sh’) AS result;

-- result --

STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/44\_Rhode\_Island"

TMPDIR="/gisdata/temp/"

UNZIPTOOL=unzip

PGPORT=5432

PGHOST=localhost

PGUSER=postgres

PGPASSWORD=yourpasswordhere

PGDATABASE=geocoder

PSQL=psql

SHP2PGSQ=shp2pgsql

wget http://www2.census.gov/geo/pvs/tiger2010st/44\_Rhode\_Island/ --no-parent --relative -- ←

recursive --level=2 --accept=zip,txt --mirror --reject=html

:

:

**请参考**

**12.1.11**

**Loader\_Generate\_Nation\_Script**

Loader\_Generate\_Nation\_Script — 生成一个指定平台的shell脚本，用于加载国家和州的数据。

**用法**

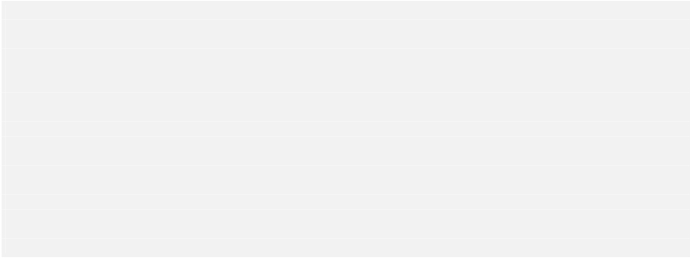
text loader\_generate\_nation\_script(text os);

**描述**

生成一个指定平台的shell脚本，用于载入表county\_all, county\_all\_lookup, state\_all 到schema为tiger\_data里面。这些表分别继承schema 为tiger下的表county, county\_lookup, state tables 。这个函数在Linux平台上使用unzip（在Windows平台上默认使用7-zip）和wget来下载数据，然后使用4.4.2 节来加载数据。注意这个函数处理的数据最小是一整个州的数据。它只会处理staging和temp下的目录。它使用下面的控制表来控制处理进程和不同操作系统的shell语法变量。

1. loader\_variables：变量表，记录不同变量的表，比如census site, year, data 和 staging schemas

2. loader\_platform ：不同平台的资料信息以及不同平台的可执行命令的位置。表的记录只带有Windows和Linux的平台的信息，可以添加更多平台的信息。



PostGIS 2.2.0dev 开发手册

593 / 686

3. loader\_lookuptables 每条记录定义了一种类型的表 (state, county),是否要处理记录中的数据以及怎样加载这种类型的数据，并且定义了每一种类型导入数据，存储数据，添加、移除列、索引和约束的步骤。每个表的名字都以相应的州为前缀，并且从schema名称为tiger的表中继承过来。比如表tiger\_data.ma\_faces继承于表tiger.faces。

可用版本: 2.1.0。

**注意**

注意如果你运行的是tiger\_2010版本，但你想要加载的是数据却是tiger\_2011版本，你需要在运行这个脚本之前，先运行drop语句 Drop\_Nation\_Tables\_Generate\_Script

**样例**

Generate script script to load nation data Windows.

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

Generate script to load up data for Linux/Unix systems.

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

**请参考**

Loader\_Generate\_Script

**12.1.12**

**Missing\_Indexes\_Generate\_Script**

Missing\_Indexes\_Generate\_Script —找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了。然后输出定义这些索引的SQL DDL语句

**用法**

text Missing\_Indexes\_Generate\_Script();

**描述**

在schema 为tiger和tiger\_data中找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了。然后输出定义这些索引的SQL DDL语句，最后执行这些脚本来添加这些缺失的索引。这个函数是很有用的函数，它添加让查询更快的新的索引，而这些索引可能在加载数据过程中丢失了。因为tiger做了一些修改提升，这个函数会自动适应新的正在使用的索引。如果这个函数什么也没有输出，这意味着你的表已经有了相关的key的索引。

可用版本: 2.0.0。

**样例**

SELECT missing\_indexes\_generate\_script();

-- output: This was run on a database that was created before many corrections were made to ←

the loading script ---

CREATE INDEX idx\_tiger\_county\_countyfp ON tiger.county USING btree(countyfp);

CREATE INDEX idx\_tiger\_cousub\_countyfp ON tiger.cousub USING btree(countyfp);

CREATE INDEX idx\_tiger\_edges\_tfidr ON tiger.edges USING btree(tfidr);



PostGIS 2.2.0dev 开发手册

594 / 686

CREATE

CREATE

CREATE

);

CREATE

);

CREATE

CREATE

INDEX idx\_tiger\_edges\_tfidl ON tiger.edges USING btree(tfidl);

INDEX idx\_tiger\_zip\_lookup\_all\_zip ON tiger.zip\_lookup\_all USING btree(zip);

INDEX idx\_tiger\_data\_ma\_county\_countyfp ON tiger\_data.ma\_county USING btree(countyfp ←

INDEX idx\_tiger\_data\_ma\_cousub\_countyfp ON tiger\_data.ma\_cousub USING btree(countyfp ←

INDEX idx\_tiger\_data\_ma\_edges\_countyfp ON tiger\_data.ma\_edges USING btree(countyfp);

INDEX idx\_tiger\_data\_ma\_faces\_countyfp ON tiger\_data.ma\_faces USING btree(countyfp);

**请参考**

Loader\_Generate\_Script, Install\_Missing\_Indexes

**12.1.13**

**Normalize\_Address**

Normalize\_Address — 根据给出的文本街道地址，返回一个复合的norm\_addy 类型值，这个值包含路的sufﬁx, preﬁx 和 type

standardized, street, streetname 等等分开的字段。这个函数只处理哪些打包在tiger\_geocoder中的数据。

**用法**

norm\_addy normalize\_address(varchar in\_address);

**描述**

根据给出的文本街道地址，返回一个复合的norm\_addy 类型值，这个值包含路的sufﬁx, preﬁx 和 typestandardized, street, streetname 等等分开的字段。这个函数只处理哪些打包在tiger\_geocoder中的数据。

把所有的地址先标准化成邮政格式的地址，是地理编码处理过程的第一步。这个函数不需要除了解码器以外的其他数据。这个函数只是使用加载在tiger 这个schema下的表tiger\_geocoder的不同direction/state/sufﬁx 的数据。因此这个函数不需要你去下载tiger的人口普查数据或者其他任何附加数据就可以使用。你可能会发现需要在tiger schema下的不同的lookup 表中添加更多的缩写或其他名称。这个函数使用不同的tiger schema下的lookup control表来标准化输入的地址。这个函数返回的norm\_addy类型的对象可以有如下顺序： ()表示这是编码必须的一个字段，而[] 表示一个可选的字段：

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

1. address是一个整型值，表示街道号。

2. predirAbbrev varchar类型，类型道路方向的前缀缩写，比如N, S, E, W。这个参数是由方向查找表direction\_lookup来控制的。

3. streetName varchar类型

4. streetTypeAbbrev varchar类型，表示街道类型的缩写，例如 St, Ave, Cir。这个是由街道类型查找表street\_type\_lookup来控制的。

5. postdirAbbrev varchar 类型道路方向的后缀缩写，比如N, S, E, W。这个参数是由方向查找表direction\_lookup来控制的。

6. internal varchar类型，内部地址表示一个公寓或者套房号码。

7. location varchar类型通常是一个城市或者一个省份。

8. stateAbbrev varchar two character US State. e.g MA, NY, MI. These are controlled by the state\_lookup table.

9. zip varchar类型，表示5位数的邮政编码，比如02109.

10. parsed boolean类型–表示地址是否是标准化的地址。函数normalize\_address在返回地址之前，把这个参数设置为true。

PostGIS 2.2.0dev 开发手册

595 / 686

**样例**

Output select ﬁelds. Use Pprint\_Addy if you want a pretty textual output.

SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev

FROM (SELECT address, normalize\_address(address) As na

FROM addresses\_to\_geocode) As g;

orig

|

streetname

| streettypeabbrev

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

28 Capen Street, Medford, MA

| Capen

| St

124 Mount Auburn St, Cambridge, Massachusetts 02138 | Mount Auburn

950 Main Street, Worcester, MA 01610 | Main

529 Main Street, Boston MA, 02129 | Main

| St

| St

| St

77 Massachusetts Avenue, Cambridge, MA 02139 | Massachusetts | Ave

25 Wizard of Oz, Walaford, KS 99912323 | Wizard of Oz

**请参考**

Geocode, Pprint\_Addy

|

**12.1.14**

**Pagc\_Normalize\_Address**

Pagc\_Normalize\_Address —根据给出的文本街道地址，返回一个复合的norm\_addy 类型值，这个值包含路的sufﬁx, preﬁx 和 type

standardized, street, streetname 等等分开的字段。这个函数只处理哪些打包在tiger\_geocoder中的lookup table表的数据。需要address\_standardizer 插件。

**用法**

norm\_addy pagc\_normalize\_address(varchar in\_address);

**描述**

根据给出的文本街道地址，返回一个复合的norm\_addy 类型值，这个值包含路的sufﬁx, preﬁx 和 typestandardized, street, streetname 等等分开的字段。把所有的地址先标准化成邮政格式的地址，是地理编码处理过程的第一步。这个函数只处理哪些打包在tiger\_geocoder中的数据。

这个函数只是使用tiger schema下的用tiger\_geocoder加载的以pagc\_为前缀的lookup表。因此这个函数不需要你去下载tiger的人口普查数据或者其他任何附加数据就可以使用。你可能会发现需要在tiger schema下的不同的lookup 表中添加更多的缩写或其他名称。这个函数使用不同的tiger schema下的lookup control表来标准化输入的地址。这个函数返回的norm\_addy类型的对象可以有如下顺序： ()表示这是编码必须的一个字段，而[] 表示一个可选的字段：

这个函数版本使用PAGC地址标准化C插件，你可以自行去下载。这个插件在类型转换和数据格式化方面有一些轻微的变化，并且有一些比较重要的功能突破。

可用版本: 2.1.0。

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

这原生的address\_standardizer 插件返回的standardaddr这时候要比norm\_addy 功能丰富一些，因为这个插件是支持国际化的地址(包括国家). standardaddr地址等价的字段如下：house\_num,predir, name, suftype, sufdir, unit, city, state, postcode

1. address是一个整型值，表示街道号。

PostGIS 2.2.0dev 开发手册

596 / 686

2. predirAbbrev varchar类型，类型道路方向的前缀缩写，比如N, S, E, W。这个参数是由方向查找表direction\_lookup来控制的。

3. streetName varchar类型

4. streetTypeAbbrev varchar类型，表示街道类型的缩写，例如 St, Ave, Cir。这个是由街道类型查找表street\_type\_lookup来控制的。

5. postdirAbbrev varchar 类型道路方向的后缀缩写，比如N, S, E, W。这个参数是由方向查找表direction\_lookup来控制的。

6. internal varchar类型，内部地址表示一个公寓或者套房号码。

7. location varchar类型通常是一个城市或者一个省份。

8. stateAbbrev varchar two character US State. e.g MA, NY, MI. These are controlled by the state\_lookup table.

9. zip varchar类型，表示5位数的邮政编码，比如02109.

10. parsed boolean类型–表示地址是否是标准化的地址。函数normalize\_address在返回地址之前，把这个参数设置为true。

**样例**

Single call example

SELECT addy.\*

FROM pagc\_normalize\_address(’9000 E ROO ST STE 999, Springfield, CO’) AS addy;

address | predirabbrev | streetname | streettypeabbrev | postdirabbrev | internal

location

| stateabbrev | zip | parsed

|

←

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

9000 | E

| ROO

| ST

|

| SUITE 999 |

←

SPRINGFIELD | CO

|

| t

Batch call. There are currently speed issues with the way postgis\_tiger\_geocoder wraps the address\_standardizer. These will

hopefully be resolved in later editions. To work around them, if you need speed for batch geocoding to call generate a normaddy

in batch mode, you are encouraged to directly call the address\_standardizer standardize\_address function as shown below which

is similar exercise to what we did in Normalize\_Address that uses data created in Geocode.

WITH g AS (SELECT address, ROW((sa).house\_num, (sa).predir, (sa).name

, (sa).suftype, (sa).sufdir, (sa).unit , (sa).city, (sa).state, (sa).postcode, true):: ←

norm\_addy As na

FROM (SELECT address, standardize\_address(’tiger.pagc\_lex’

, ’tiger.pagc\_gaz’

, ’tiger.pagc\_rules’, address) As sa

FROM addresses\_to\_geocode) As g)

SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev

FROM

g;

orig

|

streetname

| streettypeabbrev

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

529 Main Street, Boston MA, 02129 | MAIN

| ST

77 Massachusetts Avenue, Cambridge, MA 02139 | MASSACHUSETTS | AVE

25 Wizard of Oz, Walaford, KS 99912323 | WIZARD OF

|

26 Capen Street, Medford, MA

| CAPEN

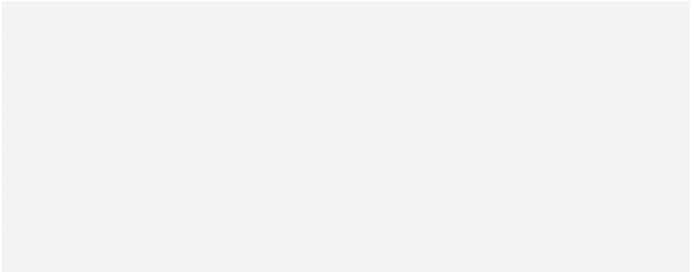
| ST

124 Mount Auburn St, Cambridge, Massachusetts 02138 | MOUNT AUBURN

950 Main Street, Worcester, MA 01610 | MAIN

| ST

| ST



PostGIS 2.2.0dev 开发手册

597 / 686

**请参考**

Normalize\_Address, Geocode

**12.1.15**

**Pprint\_Addy**

Pprint\_Addy — 根据给定的复合类型norm\_addy 对象，返回这个地址的可读性更好的展示。通常和函数normalize\_address一起使用。

**用法**

varchar pprint\_addy(norm\_addy in\_addy);

**描述**

根据给定的复合类型norm\_addy 对象，返回这个地址的可读性更好的展示。通常和函数normalize\_address一起使用。除了打包在编码器以外的数据外，不需要其他数据。这个函数通常和函数Normalize\_Address一起使用。

**样例**

Pretty print a single address

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

Pretty print address a table of addresses

SELECT address As orig, pprint\_addy(normalize\_address(address)) As pretty\_address

FROM addresses\_to\_geocode;

←

orig

|

pretty\_address

-----------------------------------------------------+----------------------------------------- -- ←

529 Main Street,

Boston MA, 02129

| 529 Main St, Boston MA, 02129

77 Massachusetts

Avenue, Cambridge, MA 02139

| 77 Massachusetts Ave, Cambridge, MA

←

02139

28 Capen Street,

Medford, MA

| 28 Capen St, Medford, MA

124 Mount Auburn

02138

St, Cambridge, Massachusetts 02138 | 124 Mount Auburn St, Cambridge, MA

←

950 Main Street,

**请参考**

Normalize\_Address

Worcester, MA 01610

| 950 Main St, Worcester, MA 01610

**12.1.16**

**Reverse\_Geocode**

Reverse\_Geocode — 输入参数是一个已知空间参考系下的point几何对象，返回值是一条包含理论上可能地址的数组和交叉街道的数组。如果参数include\_strnum\_range = true，表示把交叉街道的街道范围也算在内。

PostGIS 2.2.0dev 开发手册

598 / 686

**用法**

record Reverse\_Geocode(geometry pt, boolean include\_strnum\_range=false, geometry[] OUT intpt, norm\_addy[] OUT addy,

varchar[] OUT street);

**描述**

输入参数是一个已知空间参考系下的point几何对象，返回值是一条包含理论上可能地址的数组和交叉街道的数组。如果参数include\_strnum\_range = true，表示把交叉街道的街道范围也算在内。如果没有传递include\_strnum\_range参数值，则它的默认值是false。输出的地址是按照和目标道路的接近程度倒序排列的，因此第一个地址是最接近正确的。为什么我们说是理论的地址而不是实际的地址呢，因为Tiger 的数据并没有真正的地址，而只是街道范围。因此理论地址只是基于街道范围做的插值地址。比如像我这边地址之一进行插值后，会返回26 Court St. 和 26 Court Sq，虽然没有26 Court Sq这个地址。这是因为这个点可能是2个街道相交的地方，因此逻辑插值的地方在两个街道。该逻辑还假定地址是沿着一条街道，当然是错误的，因为你可以有一个市政建设占用了一个很好的块的街道范围内，其余的建筑物聚集在最后。

注意:这个函数依赖Tiger数据。如果你还没有加载覆盖该点的数据，那么你会返回都是NULL的一条记录。

返回记录包含的元素如下：

1. intpt ：point类型的数组，是贴金输入点的中心线的点。数量和地址数量一致。

2. addy：是一个norm\_addy类型的数组（标准化地址），这些地址是可能匹配输入点的地址数组，数组的第一个元素是最可能匹配输入点的。一般来说，数组元素应该只有一个，除非这个点恰好是2个或3个街道的交点，或者这个点不在路上或者在路边上。

3. street ：是一个varchar类型的数组。这些是有交叉的街道 (就是有相交的街道，或者街道投影的点)。

可用版本: 2.0.0。

**样例**

Example of a point at the corner of two streets, but closest to one. This is approximate location of MIT: 77 Massachusetts Ave,

Cambridge, MA 02139 注意 that although we don’t have 3 streets, PostgreSQL will just return null for entries above our upper

bound so safe to use. This includes street ranges

SELECT pprint\_addy(r.addy[1]) As st1, pprint\_addy(r.addy[2]) As st2, pprint\_addy(r.addy[3]) ←

As st3,

array\_to\_string(r.street, ’,’) As cross\_streets

FROM reverse\_geocode(ST\_GeomFromText(’POINT(-71.093902 42.359446)’,4269),true) As r ←

;

result

------

st1

| st2 | st3 |

cross\_streets

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

67 Massachusetts Ave, Cambridge, MA 02139 |

|

| 67 - 127 Massachusetts Ave,32 - 88 ←

Vassar St

Here we choose not to include the address ranges for the cross streets and picked a location really really close to a corner of 2

streets thus could be known by two different addresses.

PostGIS 2.2.0dev 开发手册

599 / 686

SELECT pprint\_addy(r.addy[1]) As st1, pprint\_addy(r.addy[2]) As st2,

pprint\_addy(r.addy[3]) As st3, array\_to\_string(r.street, ’,’) As cross\_str

FROM reverse\_geocode(ST\_GeomFromText(’POINT(-71.06941 42.34225)’,4269)) As r;

result

--------

st1

|

st2

| st3 | cross\_str

---------------------------------+---------------------------------+-----+------------------------ ←

5 Bradford St, Boston, MA 02118 | 49 Waltham St, Boston, MA 02118 |

| Waltham St

For this one we reuse our geocoded example from Geocode and we only want the primary address and at most 2 cross streets.

SELECT actual\_addr, lon, lat, pprint\_addy((rg).addy[1]) As int\_addr1,

(rg).street[1] As cross1, (rg).street[2] As cross2

FROM (SELECT address As actual\_addr, lon, lat,

reverse\_geocode( ST\_SetSRID(ST\_Point(lon,lat),4326) ) As rg

FROM addresses\_to\_geocode WHERE rating > -1) As foo;

actual\_addr

int\_addr1

|

lon

|

|

lat

|

cross1

←

|

←

cross2

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

529 Main Street, Boston MA, 02129 | -71.07181

Boston, MA 02129 | Medford St

|

| 42.38359 | 527 Main St,

←

77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428

Massachusetts Ave, Cambridge, MA 02139 | Vassar St

|

| 42.35988 | 77

←

26 Capen Street, Medford, MA

Medford, MA 02155 | Capen St

| -71.12377

| Tesla Ave

| 42.41101 | 9 Edison Ave, ←

124 Mount Auburn St, Cambridge, Massachusetts 02138 | -71.12304

Rd, Cambridge, MA 02138 | Mount Auburn St |

| 42.37328 | 3 University

←

950 Main Street, Worcester, MA 01610 | -71.82368

| 42.24956 | 3 Maywood St, ←

Worcester, MA 01603 | Main St

**请参考**

Pprint\_Addy, Geocode

| Maywood Pl

**12.1.17**

**Topology\_Load\_Tiger**

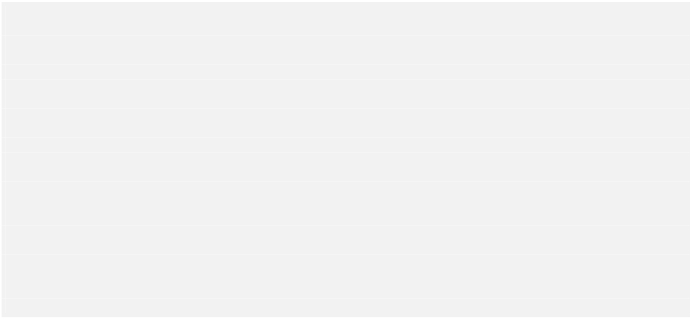
Topology\_Load\_Tiger —把tiger数据定义的区域加载到一个PostGIS Topology拓扑结构中，然后把tiger数据转换成拓扑结构所在的空间参考系中以及再把point分割到拓扑结构的tolerance容差精度。

**用法**

text Topology\_Load\_Tiger(varchar topo\_name, varchar region\_type, varchar region\_id);

**描述**

把tiger数据定义的区域加载到一个PostGIS Topology拓扑结构中，然后把faces, nodes 和 edge等对象转换到目标拓扑结构所在的空间参考系中，同时把点分割到目标拓扑的tolerance精度。新创建的faces, nodes, edges保持和原始的tiger 数据的faces, nodes, edges一样的对象ID，以便新数据集将来更容易和原始的tiger数据保持一致性。这个函数的返回值是处理过程的概述。这个函数是很有用的，比如在你想重新划分你需要的新形成的多边形或者想让生成多边形没有交叠。



PostGIS 2.2.0dev 开发手册

600 / 686

**注意**

这个函数依赖于Tiger数据以及PostGIS拓扑模块的安装。想要更多新，请参考第11章和2.4.1节。如果你还没有加载覆盖感兴趣点的数据，那么将不会创建任何拓扑结构记录。如果你使用拓扑函数没能创建成功一个拓扑结构时候，这个函数同样也会执行失败。

**注意**

大多数的拓扑验证失败都是因为转换后的边上的点没有排齐或者是有交叠导致的tolerance容差问题。如果遇到拓扑验证失败情况，想要修正这种情况，你需要提高或者降低tolerance的精度。

需要的参数：

1. topo\_name ：已经存在的PostGIS 拓扑结构名称，是要加载数据的目标拓扑。

2. region\_type ：缓冲区类型，当前只支持地方和区县，我们的开发计划是支持更多的类型。定义缓冲区的表有tiger.place, tiger.county等等。

3. region\_id ：这个与TIGER称之为geoid的等价。这是表中区域的唯一ID。对于place类型，这个字段的是只是表tiger.place的plcidfp字段值，对于county类型，这个值是表tiger.county的cntyidfp字段值。

可用版本: 2.0.0。

**Example: Boston, Massachusetts Topology**

Create a topology for Boston, Massachusetts in Mass State Plane Feet (2249) with tolerance 0.25 feet and then load in Boston

city tiger faces, edges, nodes.

SELECT topology.CreateTopology(’topo\_boston’, 2249, 0.25);

createtopology

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

15

-- 60,902 ms ~ 1 minute on windows 7 desktop running 9.1 (with 5 states tiger data loaded)

SELECT tiger.topology\_load\_tiger(’topo\_boston’, ’place’, ’2507000’);

-- topology\_loader\_tiger --

29722 edges holding in temporary. 11108 faces added. 1875 edges of faces added.

nodes added.

20576

←

19962 nodes contained in a face.

0 edge start end corrected.

31597 edges added.

-- 41 ms --

SELECT topology.TopologySummary(’topo\_boston’);

-- topologysummary--

Topology topo\_boston (15), SRID 2249, precision 0.25

20576 nodes, 31597 edges, 11109 faces, 0 topogeoms in 0 layers

-- 28,797 ms to validate yeh returned no errors --

SELECT \* FROM

topology.ValidateTopology(’topo\_boston’);

error

|

id1

|

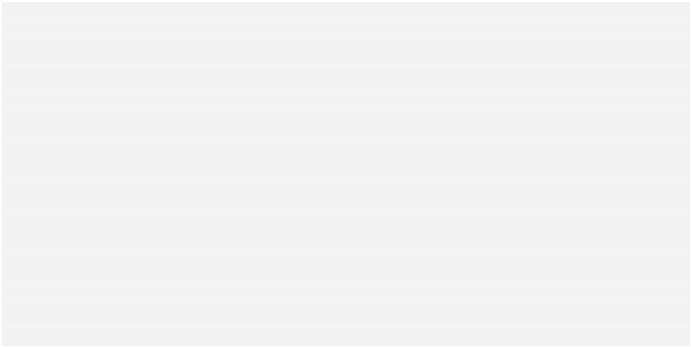
id2

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

**Example: Suffolk, Massachusetts Topology**

Create a topology for Suffolk, Massachusetts in Mass State Plane Meters (26986) with tolerance 0.25 meters and then load in

Suffolk county tiger faces, edges, nodes.



PostGIS 2.2.0dev 开发手册

601 / 686

SELECT topology.CreateTopology(’topo\_suffolk’, 26986, 0.25);

-- this took 56,275 ms ~ 1 minute on Windows 7 32-bit with 5 states of tiger loaded

-- must have been warmed up after loading boston

SELECT tiger.topology\_load\_tiger(’topo\_suffolk’, ’county’, ’25025’);

-- topology\_loader\_tiger --

36003 edges holding in temporary. 13518 faces added. 2172 edges of faces added.

24761 nodes added.

24075 nodes contained in a face.

0 edge start end corrected.

38175

←

edges added.

-- 31 ms --

SELECT topology.TopologySummary(’topo\_suffolk’);

-- topologysummary--

Topology topo\_suffolk (14), SRID 26986, precision 0.25

24761 nodes, 38175 edges, 13519 faces, 0 topogeoms in 0 layers

-- 33,606 ms to validate --

SELECT \* FROM

topology.ValidateTopology(’topo\_suffolk’);

error

|

id1

|

id2

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

coincident nodes

| 81045651 | 81064553

edge crosses node | 81045651 | 85737793

edge crosses node | 81045651 | 85742215

edge crosses node | 81045651 | 620628939

edge crosses node | 81064553 | 85697815

edge crosses node | 81064553 | 85728168

edge crosses node | 81064553 | 85733413

**请参考**

CreateTopology, CreateTopoGeom, TopologySummary, ValidateTopology

**12.1.18**

**Set\_Geocode\_Setting**

Set\_Geocode\_Setting —设置影响geocode函数的行为的setting参数。

**用法**

text Set\_Geocode\_Setting(text setting\_name, text setting\_value);

**描述**

使用存储在tiger.geocode\_settings表中的值为该函数的setting相关的参数指定值。通过settings 你可以切换函数的debugging类型，然后函数的转码的rating值就可以做到控制。settings列表参考Get\_Geocode\_Setting.

可用版本: 2.1.0。

**Example return debugging setting**

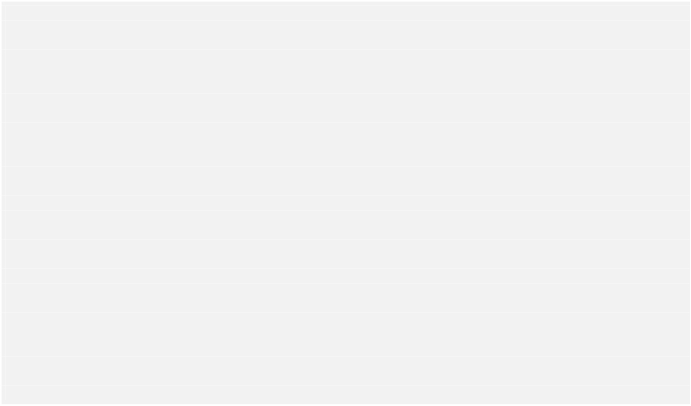
If you run Geocode when this function is true, the NOTICE log will output timing and queries.

SELECT set\_geocode\_setting(’debug\_geocode\_address’, ’true’) As result;

result

---------

true



PostGIS 2.2.0dev 开发手册

602 / 686

**请参考**

Get\_Geocode\_Setting

PostGIS 2.2.0dev 开发手册

603 / 686

**Chapter 13**

**PostGIS 特殊函数索引列表**

**13.1**

**PostGIS 聚合函数**

下面给出的空间聚合函数是有PostGIS特别提供的，这些函数可以像其他sql聚合函数比如sum，average一样使用。

· ST\_3DExtent - 一个返回包含多个geometry对象的3D bounding box的聚集函数

· ST\_Accum - 聚合函数，返回输入几何对象形成的数组

· ST\_AsTWKBAgg -聚合一些几何对象，并返回TWKB格式的描述对象

· ST\_AsTWKBAgg -聚合一些几何对象，并返回TWKB格式的描述对象

· ST\_Collect -从其他几何类型对象的collection 返回一个具体的ST\_Geometry值（对应的对象）

· ST\_Extent -一个返回包含多个geometry对象的bounding box的聚集函数

· ST\_MakeLine -根据point或line几何类型创建Linestring类型对象

· ST\_MemUnion -与函数ST\_Union功能一样，但是会使用更少的内存以及更多的处理时间

· ST\_Polygonize -聚合函数，从一个线条类的几何类型对象几何创建一个包含Polygon对象的GeometryCollection类型对象

· ST\_SameAlignment -如果上之间有相同的特征值skew, scale, spatial ref （即SRID），则返回true。如果有不同，则返回false。

· ST\_Union -返回一个代表一些几何对象的并集的几何对象\

· TopoElementArray\_Agg -返回一个topoelementarray数组，它是element\_id，type arrays (topoelements)的集合。

**13.2**

**满足SQL-MM规范的PostGIS函数**

下面给出的PostGIS函数满足SQL/MM 3 规范

**注意**

SQL-MM规范默认所有的几何对象构建函数的SRID值为0，而PostGIS默认的SRID值为-1



PostGIS 2.2.0dev 开发手册

604 / 686

· ST\_3DDWithin-如果两个三维3d(z)几何对象的三维距离指定的距离值范围内，则返回TRUE。该函数方法实现了SQL/MM规范： SQL-MM ?

· ST\_3DDistance - 对于3维的geometry类型对象，返回两个几何对象的最短笛卡尔距离(基于SRS表)，单位以投影系单位为准。

该函数方法实现了SQL/MM规范： SQL-MM ?

· ST\_3DIntersects -如果几何对象在3维空间内相交，则返回TRUE，该函数只适用于Point和LINESTRING类型。该函数方法实现了SQL/MM规范： SQL-MM 3: ?

·ST\_AddEdgeModFace -在一个face面内添加一条新的边，把这个face面分开，然后修改原始的面并添加一个新的面。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13

· ST\_AddEdgeNewFaces - 在一个face面内添加一条新边界，把这个face面分开，然后删除掉原始的面，并用两个新的面来替换。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12

· ST\_AddIsoEdge-添加一个由参数alinestring定义的孤立的边到一个拓扑结构中，用于连接两个已经存在的由参数anode和

anothernode定义的孤立点。然后返回新边的边ID。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4

·ST\_AddIsoNode -在一个拓扑结构的一个面上创建一个孤立的结点并返回新的结点的id。如果面是null空值，那么结点依然会被创建。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Net Routines: X+1.3.1

· ST\_Area -返回输入对象是polygon或者multi-polygon对象的面积。对于geometry对象来说，面积单位是SRID规定的单位，对于geography对象，面积是平方米。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.1.2, 9.5.3

· ST\_AsBinary -返回一个没有SRID信息的WKB描述的geometry或geography对象.该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.37

· ST\_AsText -返回不含有SRID信息的geometry或geography对象的WKT表示。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.25

· ST\_Boundary -返回这个几何类型对象的边界组成的闭包。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.14

· ST\_Buffer -对geometry类型对象来说，返回以给定点为中心点，距离小于或等于指定距离的所有点组成的几何对象。计算方式是按照spatial\_ref\_sys表中对该类型的几何对象的描述来计算的。对于geography类型对象：使用一个2D维度转换包装器。PostGIS 1.5版本引入了对不同的end cap和mitre参数的设置来控制几何对象的形状。buffer\_style参数控制如下：quad\_segs=#,endcap=round|ﬂat|square,join=round|mitre|bevel,mitre\_limit=#.#。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.17

· ST\_Centroid–返回一个geometry对象的几何中心。该函数方法实现了SQL/MM规范： SQL-MM 3:8.1.4, 9.5.5

· ST\_ChangeEdgeGeom -不改变拓扑结构的情况下，改变一个边的形状。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6

· ST\_Contains -ST\_Contains —当且仅当几何对象B的所有点没有在几何对象A的外部，并且B最少有一个点在A的内部（译者注：意思A把B完全包含了）。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.31

· ST\_ConvexHull-返回包含所有几何对象的最小的凸包该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.16

· ST\_CoordDim -返回ST\_ Geometry 值对应的几何类型的坐标维度值。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.3

· ST\_CreateTopoGeo -把一个几何对象的集合添加到一个指定的空拓扑结构中，然后返回处理成功的信息。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details -- X.3.18

· ST\_Crosses -如果两个输入的对象有部分交集但不是完全相交，则该函数返回TRUE。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.29

· ST\_CurveToLine -把一个CIRCULARSTRING/CURVEDPOLYGON对象转换成一个LINESTRING/POLYGON对象。该函数方法实现了规范SQL-MM 3: 7.1.7

PostGIS 2.2.0dev 开发手册

605 / 686

· ST\_Difference -根据输入的几何对象A和B，返回一个几何对象A的除了和几何对象B相交的剩余部分。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.20

· ST\_Dimension -返回输入的几何对象固有的维度值，这个维度值必须小于或等于坐标系的维度值。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.2

· ST\_Disjoint -如果两个几何对象没有空间相交则返回TRUE,如果他们没有共同的空间。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.26

· ST\_Distance -对于geometry类型对象，返回两个几何对象的2维的最小笛卡尔距离。对于geography类型对象，返回WGS84参考系两个geography对象之间的最小空间距离，距离单位是米。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.23

· ST\_EndPoint -返回一个LINESTRING的最后一个point对象。SQL-MM 3: 7.1.4

·ST\_Envelope -从提供的geometry类型中返回一个box的边界值精度为float8的geometry类型。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.15

· ST\_Equals -返回给定的几何对象是否是相同的几何对象，不考虑几何对象之间的方向差异（译者注：比如有一个几何对象，将该几何对象绕着其中一个顶点旋转20度，得到一个新的几何对象，那么这个旋转后的几何对象和原来的几何对象是相同的，

因为不考虑方向）。该函数方法实现了SQL/MM规范：SQL-MM 3: 5.1.24

· ST\_ExteriorRing -返回一个POLYGON 几何类型的外环，如果输入类型不是POLYGON类型，返回NULL值，该函数不支持MULTIPOLYGON。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.2.3, 8.3.3

· ST\_GMLToSQL -根据GML表述的对象返回一个ST\_Geometry值，这个函数是ST\_GeomFromGML的别名。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.50 (except for curves support).

· ST\_GeomCollFromText -根据WKT和给定的SRID值创建一个geometry collection类型的对象，如果SRID值没有给出，默认为0。

该函数方法实现了SQL/MM规范：

· ST\_GeomFromText -根据WKT描述返回一个具体的ST\_Geometry 值。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.40

· ST\_GeomFromWKB -根据WKB表述创建一个geometry几何类型对象，SRID可选。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.41

· ST\_GeometryFromText -根据WKT描述的对象返回一个具体的ST\_Geometry 函数值(也是一个geometry对象)，该函数是 ST\_GeomFromText的别名，即两者等价。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.40

· ST\_GeometryN -该函数的输入参数是一个collection对象和一个整数N值，返回这个collection里面的第N个对象，这些collection类型可以是GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINES，MULTICURVE or (MULTI)POLYGON, POLYHEDRALSURFACE，其他值返回为NULL。该函数方法实现了SQL/MM规范： SQL-MM 3: 9.1.5

· ST\_GeometryType -返回ST\_Geometry值对应的几何对象的几何类型。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.4

· ST\_GetFaceEdges -返回包围一个面的有序边界集合。该函数方法实现了SQL/MM规范：

SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5

· ST\_GetFaceGeometry -根据给定的拓扑结构和面ID，返回一个polygon 对象。该函数方法实现了SQL/MM规范： SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16

· ST\_InitTopoGeo -创建一个拓扑结构的schema，并将这个新schema注册到系统表the topology.topology 中，同时返回该函数处理过程的概要。该函数方法实现了SQL/MM规范： SQL-MM 3 Topo-Geo and Topo-Net 3: RoutineDetails: X.3.17

· ST\_InteriorRingN -返回一个POLYGON几何类型的第N个内环，如果输入的geometry类型不是POLYGON类型或者N超出了范围，返回NULL值。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.2.6, 8.3.5

·ST\_Intersection -返回几何对象相交的点集对应的geometry对象。对于geography类型对象，该函数会先将其转化成geometry类型对象，然后再转换成WGS84下的坐标。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.18

PostGIS 2.2.0dev 开发手册

606 / 686

· ST\_Intersects -如果Geometries/Geography在2维空间内有相交（有共同的空间部分），则返回TRUE。如果他们不相交，那么返回FALSE。对于geography类型对象—误差是0.00001米（因此许多很靠近的点会被认为是相交（即重合））。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.27

· ST\_IsClosed -如果LINESTRING是闭合的，即起点和终点重合，那么该函数返回值为TRUE，对于Polyhedral surface类型来说，它可能是闭合的。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.1.5, 9.3.3

· ST\_IsEmpty -如果输入的geometry参数是空的geometry collection或者空的polygon或者空的point类型等等，那么返回值将会是TRUE 。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.7

· ST\_IsRing -如果LINESTRING是简单、闭合的，则返回TRUE 。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.1.6

· ST\_IsSimple -如果输入的geometry类型没有不规则的几何点，比如自我相交或者自我相切，则返回值为TRUE。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.8

· ST\_IsValid -如果ST\_ Geometry返回的值是满足WKT或EWKT或WKB或EWKB描述的，那么这个函数返回值是TRUE 。译者注：返回ST\_Geometry是否是良好的格式，就是比如说一个多边形，那一定是多个点，如果只有一个点，就是有问题的。该函数方法实现了SQL/MM规范： SQL-MM3: 5.1.9

· ST\_Length -返回一个LINESTRING或MULTILINESTRING类型对象的2维长度。长度单位对于geometry类型对象来说以空间参考系规定的单位为标准，geography类型以椭圆参考系为参考系，单位是米（默认椭圆面）。该函数方法实现了SQL/MM规范： SQL-MM 3:7.1.2, 9.3.4

· ST\_LineFromText -根据WKT表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0. 该函数方法实现了SQL/MM规范： SQL-MM 3: 7.2.8

· ST\_LineFromWKB -根据WKB表述和给定的SRID创建一个LINESTRING几何类型对象。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.2.9

· ST\_LinestringFromWKB -根据WKB表述和给定的SRID创建一个geometry几何类型对象。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.2.9

· ST\_M -返回一个点的M坐标，如果输入不是point类型，返回NULL。输入值必须是point类型。该函数方法实现了SQL/MM规范。

· ST\_MLineFromText -根据WKT表述的几何对象返回ST\_MultiLineString值。该函数方法实现了SQL/MM规范：SQL-MM 3: 9.4.4

· ST\_MPointFromText -根据WKT表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0。该函数方法实现了SQL/MM规范： SQL-MM 3: 9.2.4

· ST\_MPolyFromText -根据WKT表述和给定的SRID创建一个MultiPolygon Geometry几何类型对象，如果SRID没有给定默认设置为0 。该函数方法实现了SQL/MM规范： SQL-MM 3: 9.6.4

· ST\_ModEdgeHeal -解开两个边的连接，通过删除两个边之间的结点，然后更正第一个边的信息（就是第一个边不再带有结

点了），再把第二个边删掉。这个函数返回删除结点的ID。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and

Topo-Net 3: Routine Details: X.3.9

· ST\_ModEdgeSplit -在一个已经存在的边上通过创建一个结点来分割一个边，这样就修改了原始的边并添加了一个新的边（带有结点）。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

· ST\_MoveIsoNode -把拓扑结构内的一个孤立点从一个位置移动到另一个位置，如果目标位置是一个已经存在的节点，则抛

出异常。返回该移动操作的详细信息。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Net Routines: X.3.2

· ST\_NewEdgeHeal -修复两个边界：先删除连接两个边界的结点，然后删除两个边界，然后再用与第一个边界anedge方向相同的边界替换这两个边界。该函数方法实现了SQL/MM规范： SQL-MM:Topo-Geo and Topo-Net 3: Routine Details: X.3.9

· ST\_NewEdgesSplit -分割一个边：首先在一个已经存在的边上创建一个新的结点，然后删除原有的边界，然后用两个新的边界替换原始边界。这个函数返回连接两个边界的结点。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Net Routines: X.3.8

PostGIS 2.2.0dev 开发手册

607 / 686

· ST\_NumGeometries -返回geometry类型是 GEOMETRYCOLLECTION (或 MULTI\*类型)里面包含的几何对象个数，对于单 个几何对象，返回1，其他情况返回NULL。该函数方法实现了SQL/MM规范： SQL-MM 3: 9.1.4

· ST\_NumInteriorRing -返回geometry对象里面第一个polygon里面包含的内环数（译者注：这个geometry对象里面包含多个polygon，最大的polygon在外面，该函数就是计算最大的polygon里面含有多少个子polygon） 该函数与ST\_NumInteriorRings

等价该函数方法实现了SQL/MM规范： SQL-MM 3: 8.2.5

· ST\_NumInteriorRings -返回输入几何对象里面第一个polygon对象里面的内环数，该函数支持POLYGON 和MULTIPOLYGON，如果geometry中没有polygon对象，则返回NULL。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.2.5

· ST\_NumPatches -返回 Polyhedral Surface . 对于非polyhedral 几何类型对象返回NULL值的面数。该函数方法实现了SQL/MM规范： SQL-MM 3: ?

· ST\_NumPoints -返回ST\_LineString 或 ST\_CircularString 值对应的几何对象里面包含的point对象个数。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.2.4

· ST\_OrderingEquals -如果两个输入的几何对象表示相同的几何对象，并且几何对象的点的顺序也是一致的，那么返回值为TRUE。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.43

· ST\_Overlaps -如果几何对象之间有共同的空间，相同的维度，但是互相之间没有相互包含，那么返回TRUE。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.32

· ST\_PatchN -返回至少包含一个面的geometry对象的第N个面，支持的输入类型是POLYHEDRALSURFACE, POLYHEDRALSURFACEM. 否则返回NULL值。该函数方法实现了SQL/MM规范： SQL-MM 3: ?

· ST\_Perimeter -返回ST\_Surface 或 ST\_MultiSurface (返回Polygon, Multipolygon类型对象) 值对应的geometry和geography类型对象的周长。对于geometry类型对象，返回的长度以空间参考系规定的单位做标准，对于geography类型对象，单位是米。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.1.3, 9.5.4

· ST\_Point - 根据给定的坐标值，返回ST\_Point值对应的几何类型对象，这个函数是OGC 函数ST\_MakePoint 的别名。该函数方法实现了SQL/MM规范： SQL-MM 3: 6.1.2

· ST\_PointFromText -根据WKT表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0,即未知。该函数方法实现了SQL/MM规范： SQL-MM 3: 6.1.8

· ST\_PointFromWKB - 根据WKB表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0,即未知。该函数方法实现了SQL/MM规范：SQL-MM 3: 6.1.9

· ST\_PointN -返回单个linestring 或 circular linestring的第N个point对象. 如果几何对象里面不包括LINESTRING对象，则返回 NULL。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.2.5, 7.3.5

· ST\_PointOnSurface -返回一个必然在Surface对象上的点 。该函数方法实现了规范 SQL/MM speciﬁcation. SQL-MM 3: 8.1.5, 9.5.6.根据该规范，函数 ST\_PointOnSurface支持Surface类型几何对象(POLYGONs, MULTIPOLYGONS, CURVED POLYGONS)。因此PostGIS似乎扩展了该规范所要求的对象。大多数数据库例如Oracle、DB2、ESRI SDE 似乎只支持Surface类型对象。而SQL Server 2008 和 PostGIS 一样支持所有几何类型对象

· ST\_Polygon -根据具体的linestring类型对象和SRID创建一个polygon对象。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.3.2

· ST\_PolygonFromText -根据WKT表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0,即未知。该函数方法实现了SQL/MM规范： SQL-MM 3: 8.3.6

· ST\_Relate -根据输入参数intersectionMatrixPattern的值，检查两个几何对象的内部、边界、外部是否有相交，如果有，则返回true。如果没有传递intersectionMatrixPattern参数，那么就返回表示两个几何对象关系的最大的maximum intersectionMatrixPattern值。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.25

· ST\_RemEdgeModFace -移除一个边，如果移除的边分隔了两个面，删除其中面的其中之一，然后修改另一个，让它占领原本属于两个面的空间，即占有原始的未分开的面。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.15

PostGIS 2.2.0dev 开发手册

608 / 686

· ST\_RemEdgeNewFace -移除一个边，如果移除的边分隔了两个面，删除原始的面并使用一个新的面进行替换。

该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.14

· ST\_RemoveIsoNode -删除孤立点并返回该操作的描述信息。如果点不上孤立点（比如是edge（边）起始或者终点），那么 会抛出异常。该函数方法实现了SQL/MM规范： SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3

· ST\_SRID - 返回ST\_ Geometry值对应对象的SRID值，该值必须存在spatial\_ref\_sys表中。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.5

· ST\_StartPoint - 返回LINESTRING几何对象的起点。该函数方法实现了SQL/MM规范： SQL-MM 3: 7.1.3

· ST\_SymDifference -返回输入对象A和B union后再去除A和B相交的部分。该函数是一个对称性函数，因为下面等式成立

ST\_SymDifference(A,B) = ST\_SymDifference(B,A).该函数方法实现了SQL/MM规范：SQL-MM 3: 5.1.21

· ST\_Touches -如果两个几何对象最少有一个共同点且他们的内部并不相交，那么返回TRUE。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.28

· ST\_Transform -返回一个根据给定的SRID转换成新的坐标系对象该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.6

· ST\_Union -返回一个代表一些几何对象的并集的几何对象。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.19 the z-index (elevation) when polygons are involved.

· ST\_WKBToSQL -根据WKB描述的对象参数，返回一个geometry类型对象，这个函数是ST\_GeomFromWKB的别名，但ST\_WKBToSQL这个函数没有SRID参数。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.36

· ST\_WKTToSQL -根据WKT描述的对象参数，返回一个geometry类型对象，这个函数是ST\_GeomFromWKT的别名。该函数方法实现了SQL/MM规范： SQL-MM 3: 5.1.34

· ST\_Within -如果输入几何对象A完全在几何对象B范围内，则返回TRUE 。该函数方法实现了SQL/MM规范：SQL-MM 3: 5.1.30

· ST\_X - 返回点的X坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。该函数方法实现了SQL/MM规范： SQL-MM 3: 6.1.3

· ST\_Y - 返回点的Y坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。该函数方法实现了SQL/MM规范： SQL-MM 3: 6.1.4

· ST\_Z - 返回点的Z坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。该函数方法实现了SQL/MM规范：

**13.3**

**PostGIS 支持Geography类型的函数**

如下给出的PostGIS函数和运算符，其输入或输出都geography数据类型的对象。

**注意**

下面的函数带有符号标识“ (T)”的表示这个函数不是原生的测量函数，请使用函数ST\_Transform进行转换后再进行这个函数调用。因此这些函数在处理的区域跨过国际日期变更线，北极和南极点以及覆盖多个UTM区域的大尺寸几何对象或成对大尺寸几何对象时候，函数的返回值可能不会像预期的那样。基本变换-（有利于UTM、Lambert Azimuthal (North/South)，在最坏的情况下落回墨卡托投影）

· ST\_Area -返回输入对象是polygon或者multi-polygon对象的面积。对于geometry对象来说，面积单位是SRID规定的单位，对于geography对象，面积是平方米



PostGIS 2.2.0dev 开发手册

609 / 686

· ST\_AsBinary - 返回一个没有SRID信息的WKB描述的geometry或geography对象.

· ST\_AsEWKT - 返回一个带有SRID信息的WKT描述的geometry对象.

· ST\_AsGML - 返回GML 2或GML 3版本格式描述的几何对象

· ST\_AsGeoJSON -返回一个GeoJSON格式（译者注：GeoJSON是一种类似json的格式数据，适用于地理编码）的几何对象.

· ST\_AsKML - 返回一个几何对象的KML 元素表示。该函数有几种形式，该函数默认的version值为2，默认的precision值为15

· ST\_AsSVG -根据给定geometry或geography对象，在一个SVG 数据中返回一个geometry对象。

· ST\_AsText -返回不含有SRID信息的geometry或geography对象的WKT表示。

· ST\_Azimuth -返回几何对象点A从正北方向按顺时针旋转到点B时候所需要的角度

· ST\_Buffer -对geometry类型对象来说，返回以给定点为中心点，距离小于或等于指定距离的所有点组成的几何对象。计算方式是按照spatial\_ref\_sys表中对该类型的几何对象的描述来计算的。对于geography类型对象：使用一个2D维度转换包装器。PostGIS 1.5版本引入了对不同的end cap和mitre参数的设置来控制几何对象的形状。buffer\_style参数控制如下：quad\_segs=#,endcap=round|ﬂat|square,join=round|mitre|bevel,mitre\_limit=#.#。

· ST\_CoveredBy -如果geometry或geography对象A的所有点都不在geometry或geography对象B的外部，则返回1（即TRUE）

· ST\_Covers -如果geometry或geography对象B的所有点都不在geometry或geography对象A的外部，则返回1（即TRUE）

· ST\_DWithin -如果输入的几何对象的都在以一个点为中心的指定的距离范围内则返回TRUE。对于geometry类型对象，单位以SRS表中的距离单位为标准。如果是geography类型，单位是米。默认的计算方式是use\_spheroid=true (以椭球面为参考系), 如果想更快的检索出所需要的点，使用use\_spheroid=false，该参数表示以正规的球体为参考系。

· ST\_Distance -对于geometry类型对象，返回两个几何对象的2维的最小笛卡尔距离。对于geography类型对象，返回WGS84参考系两个geography对象之间的最小空间距离，距离单位是米

· ST\_GeogFromText -从一个WKT规范描述的对象返回一个具体的geography对象

· ST\_GeogFromWKB -从一个WKB或EWKB规范描述的对象返回一个具体的geography对象

· ST\_GeographyFromText -从一个WKT规范描述的对象返回一个具体的geography对象。

· = -如果A的bounding box与B的bounding box相同，则返回TRUE。注意：输入的A和B几何对象的bounding box都是双精度

· ST\_Intersection -返回几何对象相交的点集对应的geometry对象。对于geography类型对象，该函数会先将其转化成geometry类型对象，然后再转换成WGS84下的坐标。

· ST\_Intersects -如果Geometries/Geography在2维空间内有相交（有共同的空间部分），则返回TRUE。如果他们不相交，那么返回FALSE。对于geography类型对象—误差是0.00001米（因此许多很靠近的点会被认为是相交（即重合））

· ST\_Length -返回一个LINESTRING或MULTILINESTRING类型对象的2维长度。长度单位对于geometry类型对象来说以空间参考系规定的单位为标准，geography类型以椭圆参考系为参考系，单位是米（默认椭圆面）。

· ST\_Perimeter -返回ST\_Surface 或 ST\_MultiSurface (返回Polygon, Multipolygon类型对象) 值对应的geometry和geography类型对象的周长。对于geometry类型对象，返回的长度以空间参考系规定的单位做标准，对于geography类型对象，单位是米。

· ST\_Project -返回一个输入类型为距离（单位：米）和方位角（单位：弧度）经投影转换后的Point类型

· ST\_Segmentize - 、返回一个修改后的geometry对象，让这个对象的的每一段的长度不能超过给出的最大长度。只在2D维度能进行计算。对于几何对象来说，长度单位需要在SRS表中，对于geography地理对象来说，距离单位是米 译者注：简单的说，这个函数可以把一个比较长的线段给切分，让最长的线段不超过给定的长度.

· ST\_Summary - 返回geometry对象的文本概要

·&& -如果输入对象A的2D bounding box与输入对象B的2D bounding box有交集，则返回TRUE

PostGIS 2.2.0dev 开发手册

610 / 686

**13.4**

**PostGIS 栅格支持函数**

如下给出的PostGIS函数和运算符，其输入或输出都栅格数据类型的对象。下面是按照字母排序的结果：

· Box3D -返回包围栅格“壳”的box 3d几何对象的形式。

· = -如果栅格A的bounding box与B的bounding box相同，则返回TRUE。注意：输入的栅格A和B的bounding box都是双精度

·&& -如果输入对象A的 bounding box与输入对象B的bounding box有交集，则返回TRUE

·&< -如果栅格A的缓冲区在B的左边，那么返回true

·&> -如果栅格A的缓冲区在B的左边，那么返回true

· ~= -如果栅格A的缓冲区和栅格B的缓冲区相同，则返回true。

· ST\_AddBand -根据给定的类型和初始值以及波段位置，返回一个带有新波段的栅格，如果波段位置没有指定，则新波段添加到栅格末尾

· ST\_AsBinary -返回不带要SRID元数据信息的WKB描述的栅格。

· ST\_AsGDALRaster -返回指定GDAL栅格格式的栅格瓦片。栅格格式是你的编译库所支持的其中一种。使用ST\_GDALRasters()

得到你的库所支持的格式列表。

· ST\_AsJPEG -返回指定波段的栅格瓦片为JPEG格式文件。如果没有指定波段，或者指定了但波段为1或超过3，那么只使用

第一个波段。若指定了3个波段，那么3个波段都会使用，并且映射到RGB色彩模式中。

· ST\_AsPNG -返回指定波段的栅格瓦片为PNG格式文件。如果波段中有1,3或4个波段，但没有指定波段，那么会使用所有波段。如果栅格波段又2个或者多于4个波段，并且没有指定波段，那么只会使用1个波段。波段会被映射到RGB或RGBA模式中。

· ST\_AsRaster -把PostGIS 的几何类型对象转换成PostGIS的栅格对象

· ST\_AsTIFF -返回栅格指定波段为简单的TIFF图像文件，如果没有指定波段，那么会尝试使用所有的波段。

· ST\_Aspect -获得栅格表面的坡向 (默认单位是度)。对于地形分析很有用。

· ST\_Band -根据一个已经存在的栅格，返回一个或更多的波段当做新的栅格。用已有的栅格来创建新的栅格很有用

· ST\_BandIsNoData -如果波段只设置了NODATA值，则返回true

· ST\_BandMetaData -返回指定的栅格波段的基本元数据信息，如果没有指定栅格波段，默认为1

· ST\_BandNoDataValue -返回给定栅格波段的NODATA值，如果没有给出指定波段，则默认为取1。

· ST\_BandPath -返回存储在文件系统中的波段的文件路径。如果没有指定是哪个波段，默认是波段1。

· ST\_BandPixelType -返回给定波段的像素类型，如果没有指定是哪个波段，默认是波段1。

· ST\_Clip -返回输入几何对象修改后的栅格。如果波段没有指定，那么会处理所有的波段。如果参数crop没有指定或者值为

TRUE，那么输出的栅格会被修改。

· ST\_ColorMap -根据输入的源栅格和一个指定的波段，创建一个新的最多4个8BUI（grayscale，RGB,RGBA模式）波段的栅格。如果没有指定波段，那么默认为波段1。

· ST\_Contains -当且仅当栅格rastB的所有点没有在栅格rastA的外部，并且rastB最少有一个点在rastA的内部（译者注：意思A

把B完全包含了）。

· ST\_ContainsProperly -如果输入栅格B和栅格A的内部相交，但不和A的边界（或外部）有接触，那么返回TRUE

PostGIS 2.2.0dev 开发手册

611 / 686

· ST\_ConvexHull -返回一个栅格的像素值为BandNoDataValue 的凸包几何对象。对于规则形状和非倾斜的栅格，这个 函数返回的值和函数ST\_Envelope结果一样，因此这个函数主要用于不规则形状的或倾斜的栅格对象。

· ST\_Count - 返回一个栅格给定波段的像素个数。如果没有指定波段，默认取波段1。如果参数exclude\_nodata\_value设置为true，那么这个函数只会统计像素值不为NODATA的像素个数。

· ST\_CountAgg -聚合函数。返回一个栅格给定波段的像素个数。如果没有指定波段，默认取波段1。如果参数exclude\_nodata\_value设置为true，那么这个函数只会统计像素值不为NODATA的像素个数。

· ST\_CoveredBy -如果栅格rastA没有点在栅格rastB外，则返回true。

· ST\_Covers -如果栅格rastB没有点在栅格rastA外，则返回true。

· ST\_DFullyWithin -如果栅格rastA和栅格rastB的距离完全在指定的距离内，则返回true。

· ST\_DWithin -如果栅格rastA和栅格rastB的距离在指定的距离内，则返回true。

· ST\_Disjoint -如果两个栅格rastA和栅格rastB没有空间相交，则返回true。

· ST\_DumpAsPolygons -根据一个栅格波段，返回一个geomval (geom,val) 记录集。如果没有指定波段号，那么默认取波段1。

· ST\_DumpValues -以2维数组形式，返回指定波段的值

· ST\_Envelope -返回一个多边形用于表示栅格的边界。

· ST\_FromGDALRaster -根据一个可以GDAL 支持的栅格数据，创建一个栅格

· ST\_GeoReference -返回栅格的地理参考元数据信息，有GDAL和ESRI格式，默认是GDAL格式

· ST\_HasNoBand -根据给定的波段，如果没有该波段，则返回true。如果没有指定波段，则默认指的是波段1。

· ST\_Height -返回一个栅格的以像素数为单位的高度

· ST\_HillShade -根据输入参数azimuth, altitude, brightness和 scale，返回栅格波段的山体阴影。

· ST\_Histogram -返回概括一个栅格或栅格覆盖数据分布的记录集。如果没有指定参数bins值，默认取autocomputed

· ST\_Intersection -返回一个栅格或者几何对象-像素值键值对的集合，返回结果表示两个栅格共同部分或者栅格矢量化后与几

何对象相交的结果。

· ST\_Intersects -如果栅格rastA和栅格rastB空间上相交，则返回true

· ST\_IsEmpty -如果栅格是空的(width = 0 and height = 0)，返回true，否则返回false

· ST\_MakeEmptyRaster -根据给定的维度（宽度和高度），参考坐标或 X,Y 位置（通常在栅格左上角或左下角），像元大小和旋转信息(scalex, scaley, skewx & skewy) 和SRID值。如果传递了一个栅格对象，则返回一个新的栅格，新栅格和原栅格有一样的大小、对齐方式和SRID值。如果SRID值省略了，则SRID值设置为0（未知），返回一个空栅格（没有波段的栅格）

· ST\_MapAlgebra -回调函数的版本——根据输入的一个或多个栅格，波段位置和用户指定的回调函数，返回一个单波段的栅格。

· ST\_MapAlgebraExpr -输入参数是1个波段的函数版本。根据一个用户定义在输入栅格波段和pixeltype值的PostgreSQL代数表

达式，生成一个带有一个波段的栅格。如果没有指定波段是哪个，默认取波段1。

· ST\_MapAlgebraExpr -输入参数是两个栅格波段的版本。根据一个用户定义在两个输入参数栅格波段和pixeltype值的PostgreSQL代数表达式，生成一个带有一个波段的栅格。如果没有指定每一个栅格的波段是哪个，那么默认都取波段1。输出栅格的对其方式 (scale, skew and pixel corners) 和第一个栅格参数相同，边界定义由参数"extenttype"定义，参数"extenttype"的值可以是: INTERSECTION, UNION, FIRST, SECOND。

PostGIS 2.2.0dev 开发手册

612 / 686

· ST\_MapAlgebraFct -入参数是1个波段的函数版本。根据一个用户定义在输入栅格波段和pixeltype值的PostgreSQL函数，生成一个带有一个波段的栅格。如果没有指定波段是哪个，默认取波段1。

· ST\_MapAlgebraFct - 入参数是两个栅格波段的版本。根据一个用户定义在两个输入参数栅格波段和pixeltype值的PostgreSQL函数，生成一个带有一个波段的栅格。如果没有指定每一个栅格的波段是哪个，那么默认都取波段1。 参数"extenttype"如果没有指定值，默认值是: INTERSECTION。

· ST\_MapAlgebraFctNgb -入参数是一个波段的函数版本：该函数使用用户自定义的PostgreSQL 函数。该函数会返回一个栅格，其值是一个PLPGSQL自定义函数的结果，该自定义函数与输入栅格波段的近邻值有关。

· ST\_MapAlgebra -带有SQL表达式参数expression的函数版本。根据一个或更多输入的栅格、波段以及一个或更多用户自定义的SQL表达式。

· ST\_MetaData -返回栅格对象的基本元数据信息，例如像元大小、旋转信息等等

· ST\_MinConvexHull -返回不包含NODATA像素的栅格的凸包几何对象

· ST\_NearestValue -返回给定波段的给定像元的最近的非NODATA的像元的值，给定像元可以通过行和列的位置或者栅格所在的空间参考系的几何坐标指定。

· ST\_Neighborhood -返回一个2维双精度的不包含NODATA值的像元值数组，像元是在给定波段像元周围。像元坐标用行列位置指定或几何点指定，参考系以栅格所处参考系为准。

· ST\_NotSameAlignmentReason -如果栅格对齐了那么返回描述信息，如果没有对齐，返回没有对齐的原因。

· ST\_NumBands -返回栅格对象的波段数

· ST\_Overlaps -如果栅格rastA和rastB相交，但是一个并没有完全包含另一个，则返回true。

· ST\_PixelAsCentroid -返回栅格某个像元的中心点（point geometry类型）

· ST\_PixelAsCentroids -返回一个包含一个栅格波段的中心（点几何对象）的点几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。点几何对象表示的是像元的中心。

· ST\_PixelAsPoint -返回像元左上角的point几何对象

· ST\_PixelAsPoints -返回一个包含一个栅格波段的每个像元的点几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。点几何对象的坐标是像元的左上角。

· ST\_PixelAsPolygon -返回包含指定行和列数的像素的多边形几何对象。

· ST\_PixelAsPolygons -返回一个包含一个栅格波段的每个像元的多边形几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。

· ST\_PixelHeight -返回像元的高度，单位以空间参考系所规定的几何单位为准

· ST\_PixelOfValue -根据给定的输入值，返回所有匹配的像素的columnx，rowy坐标记录

· ST\_PixelWidth -返回像元的宽度，宽度单位以空间参考系规定的为准

· ST\_Polygon -返回一个multipolygon（多边形）几何对象，该几何对象是由那些像素值不是NODATA值的并集组成的。 如果没有指定是哪个波段，那么默认取波段1。

· ST\_Quantile -计算一个栅格或者栅格覆盖的分位点。因此一个像素值可能是一个栅格的25%, 50%, 75% 的分位点。

· ST\_RasterToWorldCoord - 根据给定点的行和列的位置值，返回栅格的左上角点的几何值X和Y（即经度和维度），行和列从1开始。

· ST\_RasterToWorldCoordX -根据给定点的行和列的位置值，返回栅格的左上角点的X坐标（即经度），行和列从1开始。

· ST\_RasterToWorldCoordY -根据给定点的行和列的位置值，返回栅格的左上角点的Y坐标（即维度），行和列从1开始。

PostGIS 2.2.0dev 开发手册

613 / 686

· ST\_Reclass -创建一个新的栅格，其波段类型将由将由原始的波段类型转换而来。 如果没有指定nband值，那么默认取波段1。其他波段都不会改变。使用场景：把一个16BUI的波段转换成8BUI的波段，以便简化导出为可见的格式。

· ST\_Resample -使用一个指定的重采样算法、新维度参数（width和height）和任意一个网格角点和一个由其他栅格定义的栅格空间参考属性（由函数的形式参数ref指定）来重采样一个栅格。

· ST\_Rescale -重采样一个栅格，只调整像素的大小，新的像素值会使用NearestNeighbor (English 或 American 单词拼写方法。译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采样算法来进行计算。默认的是NearestNeighbor算法

· ST\_Resize -重新设定栅格的高度和宽度

· ST\_Reskew -重采样一个栅格，只调整像素的skew(或者说旋转参数)，新的像素值会使用NearestNeighbor (English 或 American 单词拼写方法，译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采样算法来进行计算。默认的是NearestNeighbor算法

· ST\_Rotation -返回栅格的旋转角度，单位是弧度

· ST\_Roughness -返回一个DEM模型的“粗糙程度”的栅格。

· ST\_SRID -返回栅格的SRID值

· ST\_SameAlignment -如果上之间有相同的特征值skew, scale, spatial ref （即SRID），则返回true。如果有不同，则返回false。

· ST\_ScaleX -像素块 x 方向上的像素数应输入整数值，即宽度，坐标单位以空间参考系为准

· ST\_ScaleY -像素块Y方向上的像素数应输入整数值，即高度，坐标单位以空间参考系为准

· ST\_SetBandIsNoData -把波段的isnodata标识未TRUE

· ST\_SetBandNoDataValue -把给定的波段设置为NODATA值，如果没有指定波段，则默认波段1。要标记一个波段没有NODATA值，把NODTA值设置为NULL

· ST\_SetGeoReference -单次函数调用中设置6个地理参考值。参数之间相互用空格隔开。接受的输入类型可以是GDAL或ESRI格式，默认是GDAL格式

· ST\_SetRotation -设置栅格的旋转弧度

· ST\_SetSRID -设置栅格的SRID值

· ST\_SetScale -设置像元的X和Y值，单位以空间参考系的坐标单位为准，就是设置像元的宽度和高度。

· ST\_SetSkew -设置栅格的X和Y倾斜度（或者说旋转参数）。如果只传递一个参数，那么会将X和Y值设置相同。

· ST\_SetUpperLeft -把左上角的的像元的值设置为投影的X和Y坐标。

· ST\_SetValue -根据给定波段的给定像元，设定其值，返回一个新的修改后的栅格。 波段如果没有指定是哪个，默认从1开始。

· ST\_SetValues -根据给定的栅格波段值，设置新值，返回被设定新波段值的栅格

· ST\_SkewX -返回栅格的skew X值

· ST\_SkewY -返回栅格的skew Y值。 (或者说旋转参数).

· ST\_Slope -返回一个栅格波段的坡度（默认以度为单位）。对于地形分析很有用。

· ST\_SnapToGrid –重采样一个栅格，方法是把栅格分割成由任意一个角点像素（gridx & gridy）和可选的像素大小（scalex和scaley）定义的网格。新的像素值会使用NearestNeighbor (English 或 American 单词拼写方法，译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采样算法来进行计算。默认的是NearestNeighbor算法

· ST\_Summary -返回一个栅格内容的文本描述

PostGIS 2.2.0dev 开发手册

614 / 686

· ST\_SummaryStats -返回一个栅格或者栅格覆盖的一个指定波段的概览统计值，包括：count（像素总数）, sum（像素值之和）, mean（像素平均值）, stddev（像素值标准差）, min（最小像素值）, max（最大像素值）。如果没有指定波段，默认取波段1。

· ST\_SummaryStatsAgg -聚合函数。返回一个栅格或者一个栅格集合的某个指定波段的概览统计值，包括：count（像素总数）,sum（像素值之和）, mean（像素平均值）, stddev（像素值标准差）, min（最小像素值）, max（最大像素值）。如果没有指

定波段，默认取波段1。

· ST\_TPI -返回一个带有地形位置指数的栅格。

· ST\_TRI -返回一个带有地形耐用指数的栅格。

· ST\_Tile -根据输入的栅格和输出的栅格维度，返回一个栅格集合

· ST\_Touches -如果两个栅格rastA和rastB有至少一个共同点，但内部并不相交，那么返回true。

· ST\_Transform -根据给定的投影算法，将一个栅格的一种空间参考系转换到另一个空间参考系中。提供的投影算法有

NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos ，默认是NearestNeighbor算法

· ST\_Union -取栅格瓦片的并集生成一个包含1个或更多波段的栅格。

· ST\_UpperLeftX -返回投影系中栅格的左上角X坐标

· ST\_UpperLeftY -返回投影系中栅格的左上角Y坐标

· ST\_Value -返回给定波段的行和列的像元的值。波段如果没有指定是哪个，默认从1开始。如果exclude\_nodata\_value 值为false，那么所有包含NODATA值的像元都会返回value值，如果exclude\_nodata\_value没有指定，那么这个参数值将从栅格元数

据中读取。

· ST\_ValueCount -返回一个记录集，包括像素值和指定栅格（或栅格覆盖）的指定波段的像素值在一个值集合内的像素个数。如果没有指定波段，那么默认是波段1。默认也不统计值为NODATA的像素。像素值如果不是整数，那么像素值会进行round

四舍五入处理得到一个最接近的整数值。

· ST\_Width -返回栅格的宽度，宽度以像素个数计数

· ST\_Within -当且仅当栅格rastA的所有点没有在栅格rastB的外部，并且rastA最少有一个点在rastB的内部（译者注：意思A把B完全包含了）。

· ST\_WorldToRasterCoord -根据给定X和Y值（通常是经度和维度），返回左上角的位置信息作为一条记录的行和列，或者栅格所在空间参考系中的一个点几何对象。

· ST\_WorldToRasterCoordX -返回点几何对象在栅格中的列位置，或则栅格的空间参考系统中的X和Y空间坐标。

· ST\_WorldToRasterCoordY -返回点几何对象在栅格中的行位置，或则栅格的空间参考系统中的X和Y空间坐标。

· UpdateRasterSRID -根据用户指定的表名和列名，修改栅格数据的SRID值

**13.5**

**PostGIS 支持Geometry / Geography / Raster 的Dump导出函数**

下面给出的PostGIS函数输入和输出都是一个geometry\_dump 类型对象集合或者一个元素

· ST\_DumpAsPolygons -根据一个栅格波段，返回一个geomval (geom,val) 记录集。如果没有指定波段号，那么默认取波段1。

· ST\_Intersection -返回一个栅格或者几何对象-像素值键值对的集合，返回结果表示两个栅格共同部分或者栅格矢量化后与几何对象相交的结果。

· ST\_Dump -返回一个组成输入几何对象g1的geometry\_dump (geom,path) 行集合

· ST\_DumpPoints - 返回组成一个几何对象的所有点的geometry\_dump (geom,path)记录集。

· ST\_DumpRings -返回将一个polygon对象 展开后dump出外环和内环的结果.

PostGIS 2.2.0dev 开发手册

615 / 686

**13.6**

**PostGIS Box对象处理函数**

下面给出的PostGIS函数输入或者输出都是PostGIS空间类型的box\*系列类型。box系列类型包括box2d和box3d

· Box2D -返回一个几何对象的最大扩展BOX2D对象

· Box3D -返回一个几何对象的最大扩展BOX3D对象

· Box3D -返回包围栅格“壳”的box 3d几何对象的形式。

· ST\_3DExtent -一个返回包含多个geometry对象的3D bounding box的聚集函数

· ST\_3DMakeBox -根据3D 类型的point几何类型创建BOX3D对象

· ST\_Box2dFromGeoHash -从一个GeoHash字符串返回一个BOX2D对象

· ST\_EstimatedExtent -返回一个给定的空间类型的估算box2d对象。该估算是基于对几何类型列的统计。如果没有指定

schema，使用当前current\_schema()对应的值

· ST\_Expand - 根据输入几何对象的bounding box返回一个向各个方向扩展的bounding box。输出对象的坐标精度使用双精度

· ST\_Extent -一个返回包含多个geometry对象的bounding box的聚集函数

· ST\_MakeBox2D -根据给定的point几何类型创建BOX2D对象

· ST\_XMax -返回2d或3d的bounding box 或一个几何对象（译者注：bonding box是包含一个区域的最小矩形）的X最大值

· ST\_XMin -返回bounding box或者2d或3d或geometry的X方向范围的最小值

· ST\_YMax -返回bounding box 或者2d或3d或geometry的Y方向范围的最大值.

· ST\_YMin -返回bounding box 或者2d或3d或geometry的Y方向范围的最小值.

· ST\_ZMax -返回bounding box 或者2d或3d或geometry的Z方向范围的最大值.

· ST\_ZMin -返回bounding box 或者2d或3d或geometry的Z方向范围的最小值

**13.7**

**PostGIS 支持3D对象处理的函数**

下面给出的PostGIS函数能够支持Z坐标的处理。

· AddGeometryColumn -添加一个几何类型的列到一个已经存在的表中。默认使用类型修改器来定义字段而不是约束器。可以

传递一个false逻辑值给use\_typmod参数来获取基于老的约束条件方式。

· Box3D -返回一个几何对象的最大扩展BOX3D对象

· DropGeometryColumn -从一个空间表中删除一个几何类型的列

· GeometryType -该函数以字符串方式返回几何对象的几何类型，例如’LINESTRING’, ’POLYGON’, ’MULTIPOINT’等等

· ST\_3DClosestPoint -返回3维内离g2几何对象最近的g1上的点。这个点也是g1和g2之间最短的线的第一个点

· ST\_3DDFullyWithin -如果3D对象的距离在另一个指定的距离值范围内，则返回TRUE

· ST\_3DDWithin -如果两个3d(z)几何对象的三维距离指定的距离值范围内，则返回TRUE

· ST\_3DDistance - 对于3维的geometry类型对象，返回两个几何对象的最短笛卡尔距离(基于SRS表)，单位以投影系单位为准。

· ST\_3DExtent -一个返回包含多个geometry对象的3D bounding box的聚集函数

PostGIS 2.2.0dev 开发手册

616 / 686

· ST\_3DIntersects -如果几何对象在3维空间内相交，则返回TRUE，该函数只适用于Point和LINESTRING类型

· ST\_3DLength - 返回一个LINESTRING或MULTILINESTRING类型对象的3维或2维长度。

· ST\_3DLength\_Spheroid -返回一个椭圆面的几何类型对象的长度，该函数会考虑到高度坐标，该函数只是函数 ST\_Length\_Spheroid的别名。

· ST\_3DLongestLine -返回两个几何对象之间最长的3维线段

· ST\_3DMakeBox -根据3D 类型的point几何类型创建BOX3D对象

· ST\_3DMaxDistance -对于3维的geometry类型对象，返回两个几何对象的最大笛卡尔距离(基于SRS表)，单位以投影系单位为准

· ST\_3DPerimeter -如果输入对象是polygon 或multipolygon对象，则返回3维周长。

· ST\_3DShortestLine - 返回两个3维几何对象之间最短的线段

· ST\_Accum - 聚合函数，返回输入几何对象形成的数组

· ST\_AddMeasure -在输入几何对象的起点和终点之间进行线性插值，返回一个新的带有M值的几何对象，如果输入的几何对象没有测量维度的M值，那么就自动驾驶。如果输入的几何对象有M值，那么会覆盖掉。 目前该函数只支持LINESTRINGS 和 MULTILINESTRINGS

· ST\_AddPoint - 在LINESTRING对象的某个点的位置之前添加一个点(点的位置计数从0开始)

· ST\_Afﬁne - 对一个几何对象在一个步骤中进行3d仿射变换，比如转换、旋转、放大等操作

· ST\_AsBinary - 返回一个没有SRID信息的WKB描述的geometry或geography对象.

· ST\_AsEWKB - 返回一个带有SRID信息的WKB描述的geometry对象.

· ST\_AsEWKT - 返回一个带有SRID信息的WKT描述的geometry对象.

· ST\_AsGML - 返回GML 2或GML 3版本格式描述的几何对象

· ST\_AsGeoJSON -返回一个GeoJSON格式（译者注：GeoJSON是一种类似json的格式数据，适用于地理编码）的几何对象.

· ST\_AsHEXEWKB - ST\_AsHEXEWKB — 返回HEXEWKB格式（文本表示）的geometry对象，对NDR数据使用little-endian编码，对XDR数据使用big-endian编码

· ST\_AsKML - 返回一个几何对象的KML 元素表示。该函数有几种形式，该函数默认的version值为2，默认的precision值为15

· ST\_AsX3D -以X3D xml节点元素返回返回一个几何对象，格式遵守: ISO-IEC-19776-1.2-X3DEncodings-XML

· ST\_Boundary -返回这个几何类型对象的边界组成的闭包

· ST\_Collect - 从其他几何类型对象的collection 返回一个具体的ST\_Geometry值（对应的对象）

· ST\_ConvexHull - 返回包含所有几何对象的最小的凸包

· ST\_CoordDim -返回ST\_ Geometry 值对应的几何类型的坐标维度值

· ST\_CurveToLine -把一个CIRCULARSTRING/CURVEDPOLYGON对象转换成一个LINESTRING/POLYGON对象

· ST\_DelaunayTriangles -根据输入对象的顶点对象，返回一个Delaunay 三角形化的结果。（译者注：所谓三角形化就是把输入的几何对象分成一个个三角形，有点类似微积分里面的思想）

· ST\_Difference -根据输入的几何对象A和B，返回一个几何对象A的除了和几何对象B相交的剩余部分。

· ST\_Dump - 返回一个组成输入几何对象g1的geometry\_dump (geom,path) 行集合

· ST\_DumpPoints -返回组成一个几何对象的所有点的geometry\_dump (geom,path)记录集。

· ST\_DumpRings - 返回将一个polygon对象 展开后dump出外环和内环的结果.

· ST\_EndPoint -返回一个LINESTRING的最后一个point对象。

PostGIS 2.2.0dev 开发手册

617 / 686

· ST\_ExteriorRing -返回一个POLYGON 几何类型的外环，如果输入类型不是POLYGON类型，返回NULL值，该函数不支MULTIPOLYGON

· ST\_Extrude -把一个Surface类型对象拉伸一定体积。

· ST\_FlipCoordinates -互换一个输入几何对象的X和Y坐标，对于用户将经度/维度坐标错误输入成维度/经度时候很有用，可以

用这个函数来修复它

· ST\_ForceCurve -把一个geometry对象向上转型为对应的curve 类型，如果可以的话(译者注：向上转型和Java里面的向上转型类似)

· ST\_ForceLHR -强制对输入的几何对象做左手定则转换

· ST\_ForceRHR -按照右手坐标定则对polygon的顶点进行转向

· ST\_ForceSFS -把几何对象转换成符合SFS 1.1版本规范的几何类型对象

· ST\_Force\_2D -将一个几何对象转换成2维中的对象，转换后的对象只有X和Y坐标（译者注：其他维度的坐标会被丢弃）

· ST\_Force\_3D -将几何对象转换成XYZ模式，该函数是ST\_Force3DZ别名

· ST\_Force\_3DZ -将几何对象转换成XYZ模式，该函数是ST\_Force3D别名

· ST\_Force\_4D -将一个几何对象转换为XYZM模式

· ST\_Force\_Collection -把geometry对象转换成GEOMETRYCOLLECTION类型对象

· ST\_GeomFromEWKB -从一个EWKB描述的几何对象返回一个具体的ST\_Geometry值.

· ST\_GeomFromEWKT -从一个EWKT描述的几何对象返回一个具体的ST\_Geometry值.

· ST\_GeomFromGML -根据一个geometry对象的GML描述，返回一个PostGIS的geometry类型对象

· ST\_GeomFromGeoJSON -该函数根据一个geojson描述的几何对象，生成一个PostGIS 的geometry对象

· ST\_GeomFromKML -该函数根据一个KML描述的几何对象，生成一个PostGIS 的geometry对象。

· ST\_GeometryN -该函数的输入参数是一个collection对象和一个整数N值，返回这个collection里面的第N个对象，这些collection类型可以是GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINES，MULTICURVE or (MULTI)POLYGON, POLYHEDRALSURFACE，其他值返回为NULL。

· ST\_GeometryType -返回ST\_Geometry值对应的几何对象的几何类型。

· ST\_HasArc -如果一个geometry对象或geometry collection包含一个CircularString对象，则返回TRUE

· ST\_InteriorRingN -返回一个POLYGON几何类型的第N个内环，如果输入的geometry类型不是POLYGON类型或者N超出了范

围，返回NULL值。

· ST\_InterpolatePoint -根据提供的点，返回输入几何对象离提供的点最近的点的M值

· ST\_IsClosed -如果LINESTRING是闭合的，即起点和终点重合，那么该函数返回值为TRUE，对于Polyhedral surface类型来说，它可能是闭合的。

· ST\_IsCollection - 如果参数是一个collection类型的几何对象如 (MULTI\*, GEOMETRYCOLLECTION, ...)，则返回值为TRUE

· ST\_IsPlanar -检验一个面是否是平面

· ST\_IsSimple -如果输入的geometry类型没有不规则的几何点，比如自我相交或者自我相切，则返回值为TRUE。

· ST\_Length\_Spheroid -在椭圆参考系中计算一个linestring/multilinestring的2D或3D长度，如果几何类型对象的坐标是经纬度形式，并且长度不需要重投影的话，使用该函数很有用

· ST\_LineFromMultiPoint -从一个MultiPoint几何类型中返回一个LineString类型对象.

· ST\_LineToCurve - 把一个LINESTRING/POLYGON 转换成一个CIRCULARSTRING, CURVED POLYGON类型对象。

· ST\_LineInterpolate\_Point -返回在一个LINESTRING对象上面在指定位置上插入（获取）的一个点的几何对象。第二个参数范围是0到1内的float8类型，表示被插入的点在整个LINESTRING的位置。（译者注：比如0.5是在中心位置插入，0.333可以认为是在1/3位置插入）

PostGIS 2.2.0dev 开发手册

618 / 686

· ST\_Line\_Substring -从一个LINESTRING钟返回一个从输入的LINESTRING提取指定起点和终点的片段，第二个和第三个参

数都是float8类型，值范围在0到1。译者注：该函数其实就是切割一个LINESTRING对象片段，像提取字符串一样。

· ST\_LocateBetweenElevations -返回高度值在指定范围内的几何对象（collection）。目前只支持 3D, 4D LINESTRINGS 和

MULTILINESTRINGS

· ST\_M -返回一个点的M坐标，如果输入不是point类型，返回NULL。输入值必须是point类型

· ST\_MakeLine - 根据point或line几何类型创建Linestring类型对象

· ST\_MakePoint - 创建一个2D,3DZ or 4D point 几何类型.

· ST\_MakePolygon -根据给定的闭合的LineString类型（译者注：闭合的linestring类型请参考https://msdn.microsoft.com/zh-cn/library/bb895372(v=sql.120).aspx）生成一个多边形，输入的几何类型必须是封闭的曲线。

· ST\_MakeValid -不丢弃顶点的情况下，尝试把无效的几何对象转换成有效的几何对象。

· ST\_MemUnion -与函数ST\_Union功能一样，但是会使用更少的内存以及更多的处理时间

· ST\_Mem\_Size - 返回一个几何对象占用的空间（以byte字节为单位）。

· ST\_MinkowskiSum – 求Minkowski和

· ST\_NDims -返回一个geometry类型的坐标维度，类型是small int类型，值为2,或4

· ST\_NPoints - 返回geometry的顶点个数。

· ST\_NRings - 如果输入的几何类型对象是polygon或者multi-polygon，返回环的个数。

· ST\_Node -对LINESTRING对象添加节点进行分隔

· ST\_NumGeometries -返回geometry类型是 GEOMETRYCOLLECTION (或 MULTI\*类型)里面包含的几何对象个数，对于单 个几何对象，返回1，其他情况返回NULL

· ST\_NumPatches -返回 Polyhedral Surface . 对于非polyhedral 几何类型对象返回NULL值的面数

· ST\_Orientation -确定表面方向

· ST\_PatchN -返回至少包含一个面的geometry对象的第N个面，支持的输入类型是POLYHEDRALSURFACE, POLYHEDRALSURFACEM. 否则返回NULL值

· ST\_PointFromWKB - 根据WKB表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0,即未知。

· ST\_PointN - 返回单个linestring 或 circular linestring的第N个point对象. 如果几何对象里面不包括LINESTRING对象，则返回 NULL。

· ST\_PointOnSurface - 返回一个必然在Surface对象上的点。

· ST\_Polygon - 根据具体的linestring类型对象和SRID创建一个polygon对象。

· ST\_RemovePoint -从一个LINESTRING对象中移除一个Point点，下标从0开始

· ST\_RemoveRepeatedPoints -返回一个删除重复点的几何对象

· ST\_Rotate - 返回一个几何对象以某个点为中心点，逆时针旋转指定弧度后的对象。

· ST\_RotateX -将一个几何对象绕X轴旋转指定弧度

· ST\_RotateY -将一个几何对象绕Y轴旋转指定弧度

· ST\_RotateZ -将一个几何对象绕Z轴旋转指定弧度

· ST\_Scale -通过将输入对象的坐标乘以一个系数来对对象进行缩放(缩小和放大)，例如ST\_Scale(geom, Xfactor,Yfactor, Zfactor).

· ST\_SetPoint -用给定的点替换一个LINESTRING对象的第N个点，下标从0开始

· ST\_Shift\_Longitude -读取每个几何对象的点或顶点的坐标，如果经度小于0，则加上360，如果大于0则不加。该函数返回的经纬度结果在0到360度范围内以便在平面地图内展现。

PostGIS 2.2.0dev 开发手册

619 / 686

· ST\_SnapToGrid -把输入的几何对象的所有点重新组成一个规则的栅格

· ST\_StartPoint - 返回LINESTRING几何对象的起点。

· ST\_StraightSkeleton - 从一个几何对象中计算出一个直线组成的框架。（译者注：意思就像人体是由一套骨骼组成一样的，该函数的作用就是找出构成这个几何对象的骨架部分）。

· ST\_SymDifference -返回输入对象A和B union后再去除A和B相交的部分。该函数是一个对称性函数，因为下面等式成立 ST\_SymDifference(A,B) = ST\_SymDifference(B,A).

· ST\_Tesselate -对一个Surface进行曲面三角形化切割（译者注：该函数的作用就是把各个surface面切成有三角形组成的面）

· ST\_TransScale -根据输入的缩放倍数，对一个几何对象进行缩放，缩放因子为Xfactor和Yfactor参数，该函数只支持2D对象

· ST\_Translate -把一个几何对象根据指定的输入偏移地址，移动到一个新的位置。例如ST\_Translate(geom, X,Y) 或者 ST\_Translate(geom, X, Y,Z).

· ST\_UnaryUnion -与函数ST\_Union类似，但处理geometry类型的对象。

· ST\_X - 返回点的X坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。

· ST\_XMax -返回2d或3d的bounding box 或一个几何对象（译者注：bonding box是包含一个区域的最小矩形）的X最大值

· ST\_XMin -返回bounding box或者2d或3d或geometry的X方向范围的最小值

· ST\_Y - 返回点的Y坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。

· ST\_YMax -返回bounding box 或者2d或3d或geometry的Y方向范围的最大值.

· ST\_YMin -返回bounding box 或者2d或3d或geometry的Y方向范围的最小值.

· ST\_Z - 返回点的Z坐标，如果输入参数不是一个点，返回NULL，输入必须是一个点类型。

· ST\_ZMax -返回bounding box 或者2d或3d或geometry的Z方向范围的最大值.

· ST\_ZMin -返回bounding box 或者2d或3d或geometry的Z方向范围的最小值

· ST\_Zmﬂag - 返回一个几何对象的ZM标识，值的对应关系如下，对于2d对象，ZM值为1记为0=2d,以此类推 1=3dm, 2=3dz, 3=4d。

· TG\_Equals -如果两个拓扑结构是由基本的拓扑要素组成，则返回true

· TG\_Intersects -如果两个基本的拓扑结构是由基本的拓扑要素组成，那么返回true

· UpdateGeometrySRID -更新包括一个geometry类型列,该列在geometry\_columns 元数据信息及SRID的所有信息。如果该列被添加了强制约束，强制约束会被更新为新的SRID，如果是用typmod形式的定义，类型修改符也会被更新

· geometry\_overlaps\_nd - 如果输入对象A的n-D bounding box与输入对象B的n-D bounding box有交集，则返回TRUE

**13.8**

**PostGIS 支持Curved Geometry（曲线几何对象）处理函数**

下面给出的PostGIS函数可以支持CIRCULARSTRING, CURVEDPOLYGON,和其他的曲线几何类型。

· AddGeometryColumn -添加一个几何类型的列到一个已经存在的表中。默认使用类型修改器来定义字段而不是约束器。可以

传递一个false逻辑值给use\_typmod参数来获取基于老的约束条件方式。

· Box2D -返回一个几何对象的最大扩展BOX2D对象

· Box3D -返回一个几何对象的最大扩展BOX3D对象

· DropGeometryColumn -从一个空间表中删除一个几何类型的列

· GeometryType -该函数以字符串方式返回几何对象的几何类型，例如’LINESTRING’, ’POLYGON’, ’MULTIPOINT’等等

PostGIS 2.2.0dev 开发手册

620 / 686

· PostGIS\_AddBBox -给一个几何对象添加bounding box

· PostGIS\_DropBBox -删除一个geometry类型对象的bounding box 缓存

· PostGIS\_HasBBox -如果该几何对象的bounding box已被缓存，则返回TRUE，否则返回FALSE.

· ST\_3DExtent -一个返回包含多个geometry对象的3D bounding box的聚集函数

· ST\_Accum -聚合函数，返回输入几何对象形成的数组

· ST\_Afﬁne - 对一个几何对象在一个步骤中进行3d仿射变换，比如转换、旋转、放大等操作

· ST\_AsBinary - 返回一个没有SRID信息的WKB描述的geometry或geography对象.

· ST\_AsEWKB - 返回一个带有SRID信息的WKB描述的geometry对象.

· ST\_AsEWKT - 返回一个带有SRID信息的WKT描述的geometry对象.

· ST\_AsHEXEWKB -返回HEXEWKB格式（文本表示）的geometry对象，对NDR数据使用little-endian编码，对XDR数据使用big-endian编码

· ST\_AsText -返回不含有SRID信息的geometry或geography对象的WKT表示。

· ST\_Collect -从其他几何类型对象的collection 返回一个具体的ST\_Geometry值（对应的对象）

· ST\_CoordDim -返回ST\_ Geometry 值对应的几何类型的坐标维度值。

· ST\_CurveToLine -把一个CIRCULARSTRING/CURVEDPOLYGON对象转换成一个LINESTRING/POLYGON对象。

· ST\_Distance -对于geometry类型对象，返回两个几何对象的2维的最小笛卡尔距离。对于geography类型对象，返回WGS84参考系两个geography对象之间的最小空间距离，距离单位是米。

· ST\_Dump -返回一个组成输入几何对象g1的geometry\_dump (geom,path) 行集合

· ST\_DumpPoints - 返回组成一个几何对象的所有点的geometry\_dump (geom,path)记录集。

· ST\_EstimatedExtent -返回一个给定的空间类型的估算box2d对象。该估算是基于对几何类型列的统计。如果没有指定

schema，使用当前current\_schema()对应的值

· ST\_FlipCoordinates -互换一个输入几何对象的X和Y坐标，对于用户将经度/维度坐标错误输入成维度/经度时候很有用，可以

用这个函数来修复它

· ST\_ForceSFS -把几何对象转换成符合SFS 1.1版本规范的几何类型对象

· ST\_Force\_2D -将一个几何对象转换成2维中的对象，转换后的对象只有X和Y坐标（译者注：其他维度的坐标会被丢弃）

· ST\_Force\_3D -将几何对象转换成XYZ模式，该函数是ST\_Force3DZ别名

· ST\_Force3DM -将几何对象转换成XYM模式

· ST\_Force\_3DZ -将几何对象转换成XYZ模式，该函数是ST\_Force3D别名

· ST\_Force\_4D -将一个几何对象转换为XYZM模式

· ST\_Force\_Collection -把geometry对象转换成GEOMETRYCOLLECTION类型对象

· ST\_GeoHash -返回一个几何对象的GeoHash表示。

· ST\_GeogFromWKB -从一个WKB或EWKB规范描述的对象返回一个具体的geography对象

· ST\_GeomFromEWKB -从一个EWKB描述的几何对象返回一个具体的ST\_Geometry值.

· ST\_GeomFromEWKT -从一个EWKT描述的几何对象返回一个具体的ST\_Geometry值.

PostGIS 2.2.0dev 开发手册

621 / 686

· ST\_GeomFromText -根据WKT描述返回一个具体的ST\_Geometry 值

· ST\_GeomFromWKB -根据WKB表述创建一个geometry几何类型对象，SRID可选。

· ST\_GeometryN -该函数的输入参数是一个collection对象和一个整数N值，返回这个collection里面的第N个对象，这些collection类型可以是GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINES，MULTICURVE or (MULTI)POLYGON, POLYHEDRALSURFACE，其他值返回为NULL。

· = -如果A的bounding box与B的bounding box相同，则返回TRUE。注意：输入的A和B几何对象的bounding box都是双精度

·&<| -如果A的bounding box覆盖或在B的bounding box的下侧，则返回TRUE

· ST\_HasArc -如果一个geometry对象或geometry collection包含一个CircularString对象，则返回TRUE

· ST\_IsClosed -如果LINESTRING是闭合的，即起点和终点重合，那么该函数返回值为TRUE，对于Polyhedral surface类型来说，它可能是闭合的。

· ST\_IsCollection - 如果参数是一个collection类型的几何对象如 (MULTI\*, GEOMETRYCOLLECTION, ...)，则返回值为TRUE

· ST\_IsEmpty -如果输入的geometry参数是空的geometry collection或者空的polygon或者空的point类型等等，那么返回值将会是TRUE 。

· ST\_LineToCurve - 把一个LINESTRING/POLYGON 转换成一个CIRCULARSTRING, CURVED POLYGON类型对象。

· ST\_Mem\_Size - 返回一个几何对象占用的空间（以byte字节为单位）。

· ST\_NPoints - 返回geometry的顶点个数。

· ST\_NRings - 如果输入的几何类型对象是polygon或者multi-polygon，返回环的个数。

· ST\_PointFromWKB - 根据WKB表述和给定的SRID创建一个geometry几何类型对象，如果SRID没有给定默认设置为0,即未知。

· ST\_PointN - 返回单个linestring 或 circular linestring的第N个point对象. 如果几何对象里面不包括LINESTRING对象，则返回 NULL。

· ST\_Rotate - 返回一个几何对象以某个点为中心点，逆时针旋转指定弧度后的对象。

· ST\_RotateZ -将一个几何对象绕Z轴旋转指定弧度

· ST\_SRID - 返回ST\_ Geometry值对应对象的SRID值，该值必须存在spatial\_ref\_sys表中。

· ST\_Scale -通过将输入对象的坐标乘以一个系数来对对象进行缩放(缩小和放大)，例如ST\_Scale(geom, Xfactor,Yfactor, Zfactor)。

· ST\_SetSRID -把一个几何对象设置为一个给定的整型值.

· ST\_TransScale -根据输入的缩放倍数，对一个几何对象进行缩放，缩放因子为Xfactor和Yfactor参数，该函数只支持2D对象

· ST\_Transform -返回一个根据给定的SRID转换成新的坐标系对象

· ST\_Translate -把一个几何对象根据指定的输入偏移地址，移动到一个新的位置。例如ST\_Translate(geom, X,Y) 或者 ST\_Translate(geom, X, Y,Z).

· ST\_XMax -返回2d或3d的bounding box 或一个几何对象（译者注：bonding box是包含一个区域的最小矩形）的X最大值

· ST\_XMin -返回bounding box或者2d或3d或geometry的X方向范围的最小值

· ST\_YMax -返回bounding box 或者2d或3d或geometry的Y方向范围的最大值.

· ST\_YMin -返回bounding box 或者2d或3d或geometry的Y方向范围的最小值.

· ST\_ZMax -返回bounding box 或者2d或3d或geometry的Z方向范围的最大值.

· ST\_ZMin -返回bounding box 或者2d或3d或geometry的Z方向范围的最小值

· ST\_Zmﬂag - 返回一个几何对象的ZM标识，值的对应关系如下，对于2d对象，ZM值为1记为0=2d,以此类推 1=3dm, 2=3dz, 3=4d。

PostGIS 2.2.0dev 开发手册

622 / 686

· UpdateGeometrySRID -更新包括一个geometry类型列,该列在geometry\_columns 元数据信息及SRID的所有信息。如果该列被添加了强制约束，强制约束会被更新为新的SRID，如果是用typmod形式的定义，类型修改符也会被更新

·&& -如果输入对象A的2D bounding box与输入对象B的2D bounding box有交集，则返回TRUE

·&&& -如果输入对象A的n-D bounding box与输入对象B的n-D bounding box有交集，则返回TRUE

**13.9**

**PostGIS Polyhedral Surface（多面体表面）几何对象**

下面给出的PostGIS函数，能够使用POLYHEDRALSURFACE, POLYHEDRALSURFACEM 这些几何类型对象

· Box2D -返回一个几何对象的最大扩展BOX2D对象

· Box3D -返回一个几何对象的最大扩展BOX3D对象

· GeometryType -该函数以字符串方式返回几何对象的几何类型，例如’LINESTRING’, ’POLYGON’, ’MULTIPOINT’等等

· ST\_3DClosestPoint -返回3维内离g2几何对象最近的g1上的点。这个点也是g1和g2之间最短的线的第一个点

· ST\_3DDFullyWithin -如果3D对象的距离在另一个指定的距离值范围内，则返回TRUE

· ST\_3DDWithin -如果两个3D（z）几何对象的三维距离指定的距离值范围内，则返回TRUE

· ST\_3DDistance -对于3维的geometry类型对象，返回两个几何对象的最短笛卡尔距离(基于SRS表)，单位以投影系单位为准。

· ST\_3DExtent -一个返回包含多个geometry对象的3D bounding box的聚集函数

· ST\_3DIntersects -如果几何对象在3维空间内相交，则返回TRUE，该函数只适用于Point和LINESTRING类型

· ST\_3DLongestLine -返回两个几何对象之间最长的3维线段

· ST\_3DMaxDistance -对于3维的geometry类型对象，返回两个几何对象的最大笛卡尔距离(基于SRS表)，单位以投影系单位为准。

· ST\_3DShortestLine -返回两个3维几何对象之间最短的线段

· ST\_Accum -聚合函数，返回输入几何对象形成的数组

· ST\_Afﬁne -对一个几何对象在一个步骤中进行3d仿射变换，比如转换、旋转、放大等操作

· ST\_Area -返回输入对象是polygon或者multi-polygon对象的面积。对于geometry对象来说，面积单位是SRID规定的单位，对于geography对象，面积是平方米

· ST\_AsBinary -返回一个没有SRID信息的WKB描述的geometry或geography对象.

· ST\_AsEWKB -返回一个带有SRID信息的WKB描述的geometry对象.

· ST\_AsEWKT -返回一个带有SRID信息的WKT描述的geometry对象.

· ST\_AsGML -返回GML 2或GML 3版本格式描述的几何对象

· ST\_AsX3D -以X3D xml节点元素返回返回一个几何对象，格式遵守: ISO-IEC-19776-1.2-X3DEncodings-XML

· ST\_CoordDim -返回ST\_ Geometry 值对应的几何类型的坐标维度值

· ST\_Dimension -返回输入的几何对象固有的维度值，这个维度值必须小于或等于坐标系的维度值

· ST\_Dump -返回一个组成输入几何对象g1的geometry\_dump (geom,path) 行集合

· ST\_DumpPoints -返回组成一个几何对象的所有点的geometry\_dump (geom,path)记录集。

PostGIS 2.2.0dev 开发手册

623 / 686

· ST\_Expand -根据输入几何对象的bounding box返回一个向各个方向扩展的bounding box。输出对象的坐标精度使用双精度

· ST\_Extent -一个返回包含多个geometry对象的bounding box的聚集函数

· ST\_Extrude -把一个Surface类型对象拉伸一定体积。

· ST\_FlipCoordinates -互换一个输入几何对象的X和Y坐标，对于用户将经度/维度坐标错误输入成维度/经度时候很有用，可以 用这个函数来修复它

· ST\_ForceLHR -强制对输入的几何对象做左手定则转换

· ST\_ForceRHR -按照右手坐标定则对polygon的顶点进行转向

· ST\_ForceSFS -把几何对象转换成符合SFS 1.1版本规范的几何类型对象

· ST\_Force2D -将一个几何对象转换成2维中的对象，转换后的对象只有X和Y坐标（译者注：其他维度的坐标会被丢弃）

· ST\_Force3D -将几何对象转换成XYZ模式，该函数是ST\_Force3DZ别名

· ST\_Force3DZ -将几何对象转换成XYZ模式，该函数是ST\_Force3D别名.

· ST\_ForceCollection -把geometry对象转换成GEOMETRYCOLLECTION类型对象

· ST\_GeomFromEWKB -从一个EWKB描述的几何对象返回一个具体的ST\_Geometry值.

· ST\_GeomFromEWKT -从一个EWKT描述的几何对象返回一个具体的ST\_Geometry值.

· ST\_GeomFromGML -根据一个geometry对象的GML描述，返回一个PostGIS的geometry类型对象

· ST\_GeometryN -该函数的输入参数是一个collection对象和一个整数N值，返回这个collection里面的第N个对象，这些

collection类型可以是GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINES，MULTICURVE or (MULTI)POLYGON,

POLYHEDRALSURFACE，其他值返回为NULL

· ST\_GeometryType -返回ST\_Geometry值对应的几何对象的几何类型

· = -如果A的bounding box与B的bounding box相同，则返回TRUE。注意：输入的A和B几何对象的bounding box都是双精度

·&<| -如果A的bounding box覆盖或在B的bounding box的下侧，则返回TRUE

· ~= -如果A的bounding box和B的bounding box相同，则返回TRUE

· ST\_IsClosed -如果LINESTRING是闭合的，即起点和终点重合，那么该函数返回值为TRUE，对于Polyhedral surface类型来

说，它可能是闭合的

· ST\_IsPlanar -检验一个面是否是平面

· ST\_Mem\_Size -返回一个几何对象占用的空间（以byte字节为单位）

· ST\_MinkowskiSum -求 Minkowski （闵可夫斯基）和

· ST\_NPoints -返回geometry的顶点个数.

· ST\_NumGeometries -返回geometry类型是 GEOMETRYCOLLECTION (或 MULTI\*类型)里面包含的几何对象个数，对于单 个几何对象，返回1，其他情况返回NULL

· ST\_NumPatches -返回 Polyhedral Surface . 对于非polyhedral 几何类型对象返回NULL值. 的面数

· ST\_Orientation -确定表面方向

· ST\_PatchN -返回至少包含一个面的geometry对象的第N个面，支持的输入类型是POLYHEDRALSURFACE, POLYHEDRALSURFACEM. 否则返回NULL值

· ST\_RemoveRepeatedPoints -返回一个删除重复点的几何对象

· ST\_Rotate -返回一个几何对象以某个点为中心点，逆时针旋转指定弧度后的对象.

PostGIS 2.2.0dev 开发手册

624 / 686

· ST\_RotateX -将一个几何对象绕X轴旋转指定弧度

· ST\_RotateY -将一个几何对象绕Y轴旋转指定弧度

· ST\_RotateZ -将一个几何对象绕Z轴旋转指定弧度

· ST\_Scale -通过将输入对象的坐标乘以一个系数来对对象进行缩放(缩小和放大)，例如ST\_Scale(geom, Xfactor,Yfactor, Zfactor).

· ST\_Shift\_Longitude -读取每个几何对象的点或顶点的坐标，如果经度小于0，则加上360，如果大于0则不加。该函数返回的经纬度结果在0到360度范围内以便在平面地图内展现

· ST\_StraightSkeleton -从一个几何对象中计算出一个直线组成的框架。（译者注：意思就像人体是由一套骨骼组成一样的，该函数的作用就是找出构成这个几何对象的骨架部分）

· ST\_Tesselate -对一个Surface进行曲面三角形化切割（译者注：该函数的作用就是把各个surface面切成有三角形组成的面）

· ST\_Transform -返回一个根据给定的SRID转换成新的坐标系对象

·&& -如果输入对象A的2D bounding box与输入对象B的2D bounding box有交集，则返回TRUE

·&&& -如果输入对象A的n-D bounding box与输入对象B的n-D bounding box有交集，则返回TRUE

**13.10**

**PostGIS 函数支持矩阵**

以下PostGIS函数是按照字母排序的空间特殊函数，以及关于这些函数处理的空间类型和他们对OGC/SQL标准规范的的兼容性说明

· A

· A

表示函数原生支持类型或子类型。

表示可以使用，但是需要进行内置的参考系转换，先转换到一个”最合适的SRID”，然后再转换回去。对于大型面要素或者面中的孔而言，这个结果可能不是所期待的，并且可能导致累积的浮点型的point对象垃圾。

.

· A

表示这个函数可以处理这种类型，因为函数内部进行了自动转换再处理, 比如自动转换成box3d进行处理，而不是

直接处理这种类型。

· A

· A

意味着这个函数只有在PostGIS添加了SFCGAL支持情况下才能支持。

意味着如果PostGIS安装时候添加了SFCGAL编译支持，那么这个函数会又SFCGAL支持，否则的话由GEOS或者内置支持。

下面单元格单词缩写表示的含义：

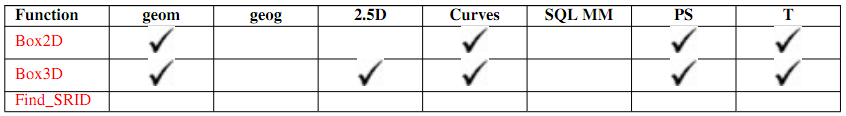
· geom – 基本的2D geometry对象，支持(x,y)形式坐标。

· geog – 基本的2D geography对象，支持(x,y)形式坐标。

· 2.5D – 在3D/4D空间中（有Z或M坐标）的基本的2D geometry对象。

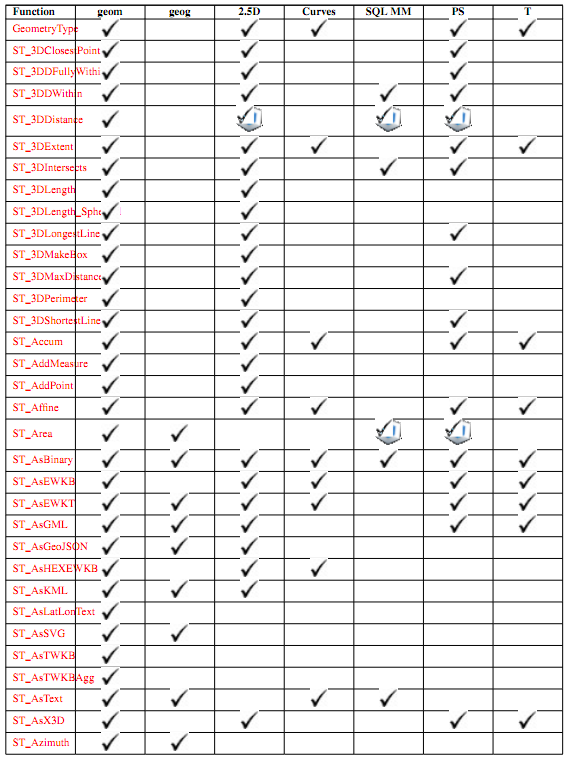
· PS – Polyhedral surfaces（多面体表面）

· T - Triangles 和 Triangulated Irregular Network surfaces (TIN)对象



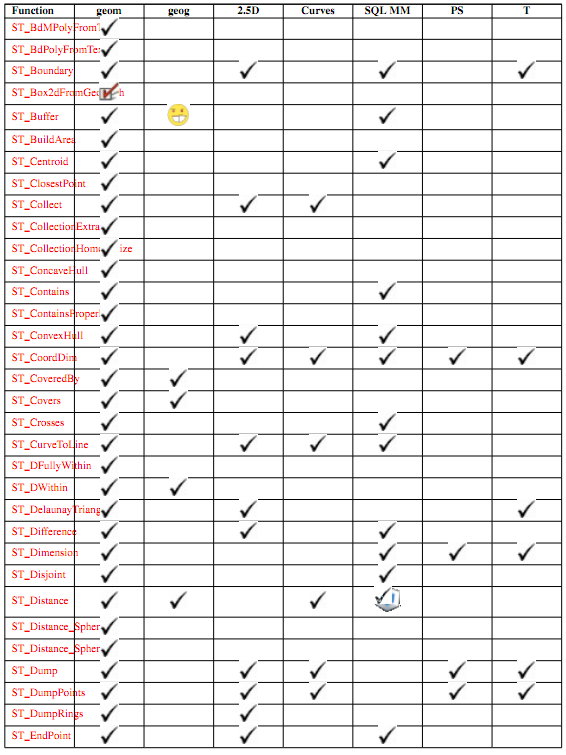
PostGIS 2.2.0dev 开发手册

625 / 686



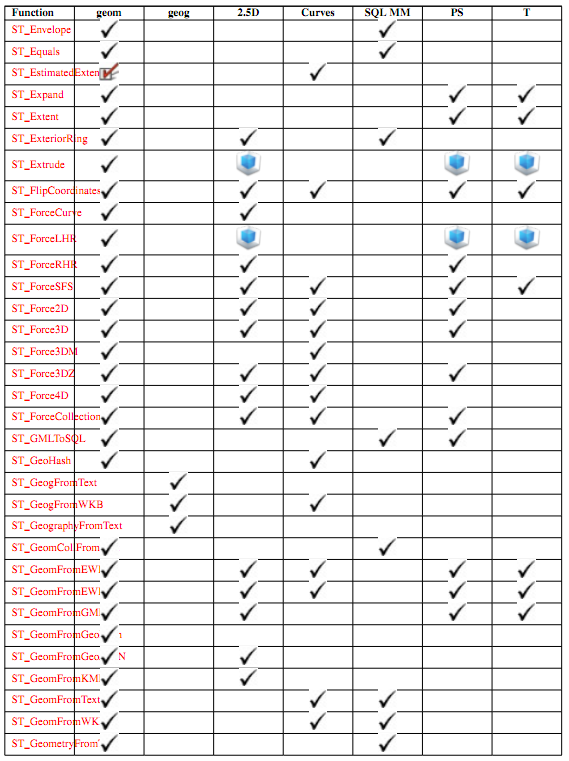
PostGIS 2.2.0dev 开发手册

626 / 686



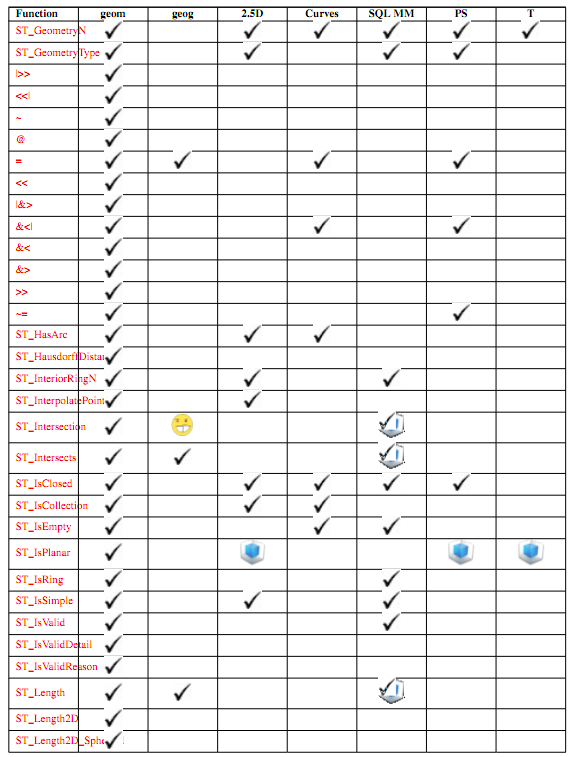
PostGIS 2.2.0dev 开发手册

627 / 686



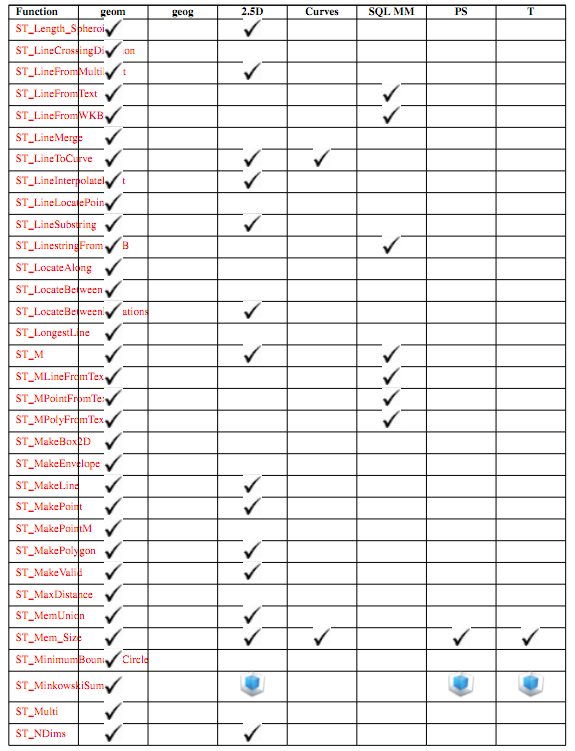
PostGIS 2.2.0dev 开发手册

628 / 686



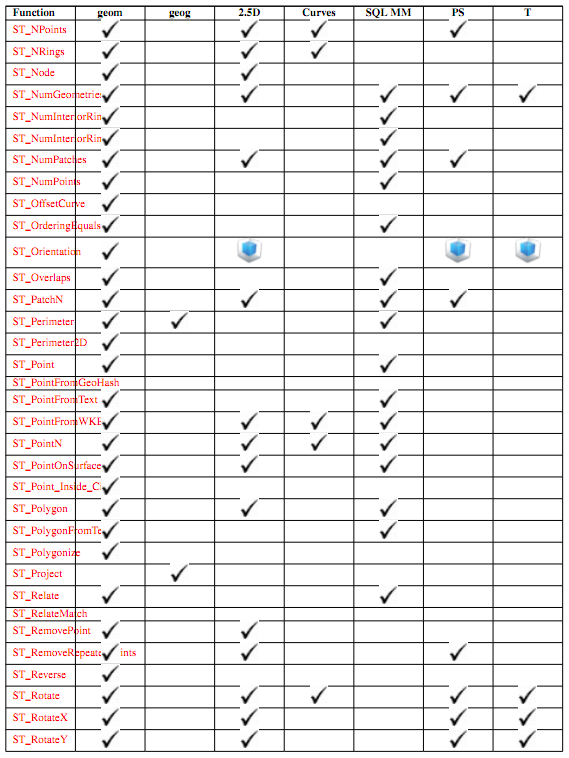
PostGIS 2.2.0dev 开发手册

629 / 686



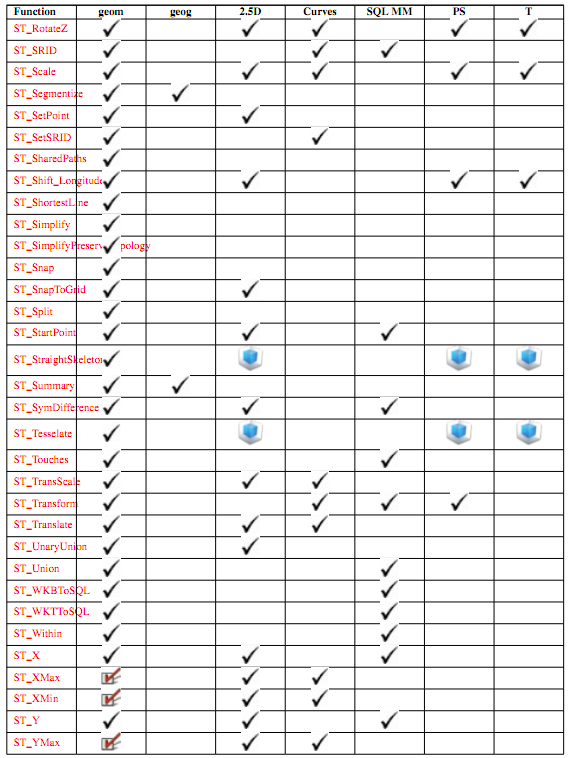
PostGIS 2.2.0dev 开发手册

630 / 686



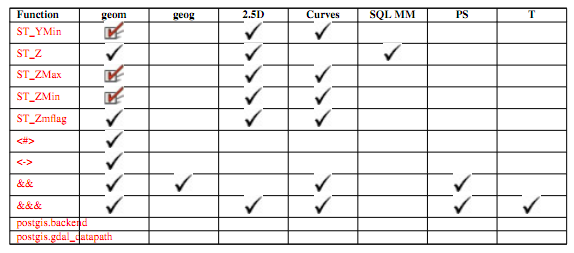
PostGIS 2.2.0dev 开发手册

631 / 686



PostGIS 2.2.0dev 开发手册

632 / 686



**13.11**

**13.11.1**

**新增加的，性能/功能提升的或改变了的PostGIS函数**

**PostGIS 2.2版本新增的或有所提升的函数**

下面给出的PostGIS函数，是新增的或者有提升的

**注意**

升级Tiger Geocoder以便支持TIGER 2013人口普查数据，请参考12.1获取更多细节

PostGIS 2.2中新添加的函数：

· ST\_AsTWKB - 可用版本: 2.2.0输出TWKB（即Tiny WKB）描述的几何对象

· ST\_AsTWKBAgg - 可用版本: 2.2.0。聚合一些几何对象，并返回TWKB格式的描述对象

· ST\_CountAgg - 可用版本: 2.2.0 。聚合函数。返回一个栅格给定波段的像素个数。如果没有指定波段，默认取波段1。如果参数exclude\_nodata\_value设置为true，那么这个函数只会统计像素值不为NODATA的像素个数。

· ST\_ForceCurve - 可用版本: 2.2.0。把一个geometry对象向上转型为对应的curve 类型，如果可以的话(译者注：向上转型和Java里面的向上转型类似)

· ST\_SummaryStatsAgg - 可用版本: 2.2.0 。聚合函数。返回一个栅格或者一个栅格集合的某个指定波段的概览统计值，包括：count（像素总数）, sum（像素值之和）, mean（像素平均值）, stddev（像素值标准差）, min（最小像素值）, max（最大像素值）。如果没有指定波段，默认取波段1。

·postgis.gdal\_datapath - 可用版本: 2.2.0 。用来指定GDAL的GDAL\_DATA值。如果没有设置，运行环境将设置GDAL\_DATA这个变量已被使用

**13.11.2**

**PostGIS 2.1版本新增或有所提升的函数**

下面给出的PostGIS函数，是新增的或者有提升的

**注意**

更多关于拓扑特性的性能提升，请参考第11章获取更多细节



PostGIS 2.2.0dev 开发手册

633 / 686

**注意**

关于bug修复 (特别是关于处理数据库之外的栅格波段),许多新函数 (通常使用一个普通的工作你可以使用更简练的代码)以及关于栅格特性的大量性能提升，请参考第9章获取更多细节。

**注意**

已经升级的Tiger Geocoder可以TIGER 2012人口普查数据，geocode\_settings添加了debugging和tweaking rating两种新选择，tiger load工具需要更少的要求，现在只需要下载必须要的加载数据，相关请参考12.1获得更多新

PostGIS 2.1版本新增的函数

· AsTopoJSON - 可用版本: 2.1.0。返回描述一个拓扑结构的TopoJSON格式描述

· Drop\_Nation\_Tables\_Generate\_Script - 可用版本: 2.1.0。生成一个SQL脚本，这个脚本用于drop掉指定schema的所有表。生成的SQL脚本drop的表从county\_all开始drop。

· Get\_Geocode\_Setting - 可用版本: 2.1.0。返回表tiger.geocode\_settings某个setting\_name的setting值。

· Loader\_Generate\_Nation\_Script- 可用版本: 2.1.0。生成一个指定平台的shell脚本，用于加载国家和州的数据。

· Pagc\_Normalize\_Address - 可用版本: 2.1.0。Pagc\_Normalize\_Address —根据给出的文本街道地址，返回一个复合的norm\_addy 类型值，这个值包含路的sufﬁx, preﬁx 和 type standardized, street, streetname 等等分开的字段。这个函数只处理哪些打包在tiger\_geocoder中的lookup table表的数据。需要address\_standardizer 插件。

· ST\_Box2dFromGeoHash - 可用版本: 2.1.0。从一个GeoHash字符串返回一个BOX2D对象

· ST\_ColorMap - 可用版本: 2.1.0。根据输入的源栅格和一个指定的波段，创建一个新的最多4个8BUI（grayscale，RGB,RGBA。模式）波段的栅格。如果没有指定波段，那么默认为波段1。

· ST\_Contains - 可用版本: 2.1.0。当且仅当栅格rastB的所有点没有在栅格rastA的外部，并且rastB最少有一个点在rastA的内部（译者注：意思A把B完全包含了）。

· ST\_ContainsProperly - 可用版本: 2.1.0。如果输入栅格B和栅格A的内部相交，但不和A的边界（或外部）有接触，那么返回TRUE

· ST\_CoveredBy - 可用版本: 2.1.0。如果栅格rastA没有点在栅格rastB外，则返回true。

· ST\_Covers - 可用版本: 2.1.0。如果栅格rastB没有点在栅格rastA外，则返回true。

· ST\_DFullyWithin - 可用版本: 2.1.0。如果栅格rastA和栅格rastB的距离完全在指定的距离内，则返回true。

· ST\_DWithin - 可用版本: 2.1.0。如果栅格rastA和栅格rastB的距离在指定的距离内，则返回true。

· ST\_DelaunayTriangles - 可用版本: 2.1.0。 - 需要GEOS版本>= 3.4.0。根据输入对象的顶点对象，返回一个Delaunay 三角形化的结果。（译者注：所谓三角形化就是把输入的几何对象分成一个个三角形，有点类似微积分里面的思想）

· ST\_Disjoint - 可用版本: 2.1.0。如果两个栅格rastA和栅格rastB没有空间相交，则返回true。

· ST\_DumpValues - 可用版本: 2.1.0。以2维数组形式，返回指定波段的值

· ST\_FromGDALRaster - 可用版本: 2.1.0。根据一个可以GDAL 支持的栅格数据，创建一个栅格

· ST\_GeomFromGeoHash - 可用版本: 2.1.0。从一个GeoHash型字符串返回一个geometry对象类型.

·ST\_InvDistWeight4ma - 可用版本: 2.1.0。栅格处理函数，它从像素的邻域计算一个像素插值。



PostGIS 2.2.0dev 开发手册

634 / 686

· ST\_MapAlgebra - 可用版本: 2.1.0。ST\_回调函数的版本——根据输入的一个或多个栅格，波段位置和用户指定的回调函数，返回一个单波段的栅格。

· ST\_MapAlgebra - 可用版本: 2.1.0。带有SQL表达式参数expression的函数版本。根据一个或更多输入的栅格、波段以及一个或更多用户自定义的SQL表达式。

· ST\_MinConvexHull - 可用版本: 2.1.0。返回不包含NODATA像素的栅格的凸包几何对象

· ST\_MinDist4ma - 可用版本: 2.1.0。栅格处理函数，返回感兴趣的像素和一个邻域像素之间的最小距离（距离单位是像素的个数）

· ST\_NearestValue -可用版本: 2.1.0。返回给定波段的给定像元的最近的非NODATA的像元的值，给定像元可以通过行和列的位置或者栅格所在的空间参考系的几何坐标指定。

· ST\_Neighborhood -可用版本: 2.1.0。返回一个2维双精度的不包含NODATA值的像元值数组，像元是在给定波段像元周围。像元坐标用行列位置指定或几何点指定，参考系以栅格所处参考系为准。

· ST\_NotSameAlignmentReason - 可用版本: 2.1.0。如果栅格对齐了那么返回描述信息，如果没有对齐，返回没有对齐的原因。

· ST\_Overlaps - 可用版本: 2.1.0。如果栅格rastA和rastB相交，但是一个并没有完全包含另一个，则返回true。

· ST\_PixelAsCentroid - 可用版本: 2.1.0。返回栅格某个像元的中心点（point geometry类型）

· ST\_PixelAsCentroids - 可用版本: 2.1.0。返回一个包含一个栅格波段的中心（点几何对象）的点几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。点几何对象表示的是像元的中心。

· ST\_PixelAsPoint - 可用版本: 2.1.0。返回像元左上角的point几何对象

· ST\_PixelAsPoints - 可用版本: 2.1.0。返回一个包含一个栅格波段的每个像元的点几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。点几何对象的坐标是像元的左上角。

· ST\_PixelOfValue - 可用版本: 2.1.0。根据给定的输入值，返回所有匹配的像素的columnx，rowy坐标记录

· ST\_PointFromGeoHash - 可用版本: 2.1.0。根据GeoHash字符串返回一个point几何类型对象

· ST\_RasterToWorldCoord - 可用版本: 2.1.0。根据给定点的行和列的位置值，返回栅格的左上角点的几何值X和Y（即经度和维度），行和列从1开始。

· ST\_Resize - 可用版本: 2.1.0。需要GDAL库 1.6.1+版本。重新设定栅格的高度和宽度

· ST\_Roughness - 可用版本: 2.1.0。返回一个DEM模型的“粗糙程度”的栅格。

· ST\_SetValues - 可用版本: 2.1.0。根据给定的栅格波段值，设置新值，返回被设定新波段值的栅格

· ST\_Simplify - 可用版本: 2.1.0。 使用Douglas-Peucker 算法，将一个拓扑结构简化成简单的几何对象

· ST\_Summary - 可用版本: 2.1.0。返回一个栅格内容的文本描述

· ST\_TPI - 可用版本: 2.1.0。返回一个带有地形位置指数的栅格。

· ST\_TRI - 可用版本: 2.1.0。返回一个带有地形耐用指数的栅格。

· ST\_Tile - 可用版本: 2.1.0。根据输入的栅格和输出的栅格维度，返回一个栅格集合

· ST\_Touches - 可用版本: 2.1.0。如果两个栅格rastA和rastB有至少一个共同点，但内部并不相交，那么返回true。

· ST\_Union - 可用版本: 2.1.0。引入了函数变体ST\_Union(rast, unionarg)。取栅格瓦片的并集生成一个包含1个或更多波段的栅格。

· ST\_Within - 可用版本: 2.1.0。当且仅当栅格rastA的所有点没有在栅格rastB的外部，并且rastA最少有一个点在rastB的内部（译者注：意思A把B完全包含了）。

PostGIS 2.2.0dev 开发手册

635 / 686

· ST\_WorldToRasterCoord - 可用版本: 2.1.0。根据给定X和Y值（通常是经度和维度），返回左上角的位置信息作为一条记录的行和列，或者栅格所在空间参考系中的一个点几何对象。

· Set\_Geocode\_Setting - 可用版本: 2.1.0。 设置影响geocode函数的行为的setting参数。

· UpdateRasterSRID - 可用版本: 2.1.0。 根据用户指定的表名和列名，修改栅格数据的SRID值

· clearTopoGeom - 可用版本: 2.1 清除拓扑结构的内容

· postgis.backend - 可用版本: 2.1.0。 该变量用于服务端程序选择geos或sfcgal库. 变量值选项: geos 或 sfcgal. 默认是geos.

下面给出的PostGIS函数是在PostGIS2.1中有提升的函数

· ST\_AddBand - 版本提升： 2.1.0 添加对addbandarg支持

· ST\_AddBand -版本提升： 2.1.0添加对库外波段支持，及形式7的支持

· ST\_AsBinary -版本提升： 2.1.0添加了参数outasin支持

· ST\_Aspect -版本提升： 2.1.0 使用函数ST\_MapAlgebra()计算，添加函数可选参数interpolate\_nodata

· ST\_Clip - 版本提升： 2.1.0 提升了性能 (完全基于C实现了这个函数).

· ST\_Distinct4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_HillShade - 版本提升： 2.1.0 利用函数 ST\_MapAlgebra() 计算，并且添加了可选参数interpolate\_nodata

· ST\_Max4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_Mean4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_Min4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_PixelAsPolygons - 版本提升： 2.1.0 添加参数exclude\_nodata\_value optional 支持

· ST\_Polygon - 版本提升： 2.1.0提高了性能(完全基于C重新实现该函数) ，并且保证返回的多边形几何是有效的。

· ST\_Range4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_SameAlignment -版本提升： 2.1.0 添加聚合版本的函数变体

· ST\_SetGeoReference - 版本提升： 2.1.0 添加了函数形式ST\_SetGeoReference(raster, double precision, ...)

· ST\_SetValue - 版本提升： 2.1.0 ST\_SetValue() 现在支持包括点在内的任意几何类型对象。几何参数geom是一个ST\_SetValues()变量geomval[]的包装。

· ST\_Slope -版本提升： 2.1.0 使用函数ST\_MapAlgebra() 并添加可选参数units, scale, interpolate\_nodata

· ST\_StdDev4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_Sum4ma -提升：2.1.0 版本添加了函数变体2的实现。

· ST\_Transform -版本提升： 2.1.0 添加函数变体支持ST\_Transform(rast, alignto)

· ST\_Union - 版本提升： 2.1.0 提升了性能 (完全基于C实现了这个函数).

· ST\_Union -版本提升： 2.1.0修改了函数ST\_Union(rast) (variant 1)的实现，把所有输入的栅格的波段都union了起来，之前的版本只union了第一个波段。

· ST\_Union - 版本提升： 2.1.0 修改了函数ST\_Union(rast, uniontype) (variant 4) 的实现，把所有输入的栅格的波段都union了起来，之前的版本只union了第一个波段。

· ST\_AsGML - 版本提升: 2.1.0 引入对于GML3版本的id支持

· ST\_Boundary - 提升: 2.1.0 版本引入对于Triangle类型的支持

· ST\_DWithin - 版本提升: 2.1.0 提高处理geography对象的速度. 参考资料 Making Geography faster.

PostGIS 2.2.0dev 开发手册

636 / 686

· ST\_DWithin - 提升：2.1.0版本引入对curve 几何对象的支持。

· ST\_Distance -版本提升: 2.1.0 提高处理geography对象的速度. 参考 Making Geography faster 获取更多细节。

· ST\_Distance -提升：2.1.0版本引入对curve 几何对象的支持。

· ST\_DumpPoints -版本提升: 2.1.0 速度更快，用C重新实现了该函数 。

· ST\_MakeValid -版本提升: 2.1.0 添加了对GEOMETRYCOLLECTION 和 MULTIPOINT.的支持

· ST\_Segmentize - Enhanced: 2.1.0 引入支持geography类型对象.

· ST\_Summary -提升: 2.1.0 添加了S标识用来表示空间参考坐标系

· toTopoGeom -版本提升： 2.1.0 添加支持处理已经存在TopoGeometry对象功能

**13.11.3**

**PostGIS 2.1版本有重大变更的函数**

下面给出的PostGIS函数在PostGIS 2.1版本可能有重大的变更。如果你使用了下面的函数，你可能需要重新检查一下已有的代码。

· ST\_Aspect -版本变更：2.1.0 在早先的版本，返回的值是以弧度为单位的，现在返回的值默认是以度为单位的。

· ST\_HillShade -版本变更：2.1.0 在早前的版本中，参数azimuth 和 altitude使用弧度表示，现在azimuth 和 altitude 都使用度来表示。

· ST\_Intersects -版本变更： 2.1.0 函数变体ST\_Intersects(raster, geometry) 的结果返回已经改变了以便匹配ST\_Intersects(geometry, raster)的返回结果

· ST\_PixelAsCentroids - 版本变更：2.1.1 ，修改了exclude\_nodata\_value值对应的操作方式。

· ST\_PixelAsPoints - 版本变更：2.1.1 ，修改了exclude\_nodata\_value值对应的操作方式。

· ST\_PixelAsPolygons - 版本变更：2.1.1 ，修改了exclude\_nodata\_value值对应的操作方式。

· ST\_Polygon -版本变更：2.1.0在之前的版本会有时候返回polygon几何对象，现在改成总是返回multipolygon几何对象。

· ST\_RasterToWorldCoordX - 版本变更：2.1.0，在之前的版本，这个函数叫做ST\_Raster2WorldCoordX

· ST\_RasterToWorldCoordY - 版本变更：2.1.0，在之前的版本，这个函数叫做ST\_Raster2WorldCoordY

· ST\_Resample -版本变更：2.1.0 参数SRID被移除了。带有引用参考栅格的函数变体不再使永引用栅格的SRID值，使用ST\_Transform() 来重投影栅格。支持无SRID的栅格.

· ST\_Rescale - 版本变更：2.1.0 支持没有SRID的栅格。

· ST\_Reskew - 版本变更：2.1.0 支持没有SRID的栅格。

· ST\_Slope -版本变更：2.1.0 在早期的版本，返回的值单位是radians（弧度），现在默认返回的是degrees（度）

· ST\_SnapToGrid - 版本变更：2.1.0 支持没有SRID的栅格。

· ST\_WorldToRasterCoordX - 版本变更：2.1.0，在之前的版本，这个函数叫做ST\_World2RasterCoordX

· ST\_WorldToRasterCoordY - 版本变更：2.1.0，在之前的版本，这个函数叫做ST\_World2RasterCoordY

· ST\_EstimatedExtent -版本变更: 2.1.0 2.0.x版本之前，该函数被称作ST\_Estimated\_Extent.

· ST\_Force2D - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_2D.

· ST\_Force3D - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_3D.

· ST\_Force3DM - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_3DM.

· ST\_Force3DZ - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_3DZ.

PostGIS 2.2.0dev 开发手册

637 / 686

· ST\_Force4D - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_4D.

· ST\_ForceCollection - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Force\_Collection.

· ST\_LineInterpolatePoint - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_LineInterpolatePoint.

· ST\_LineLocatePoint - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_Line\_Locate\_Point.

· ST\_LineSubstring - 版本变更: 2.1.0 2.0.x版本之前，该函数被称作 ST\_LineSubstring.

· ST\_Segmentize - 版本变更: 2.1.0 引入对geography类型对象支持:该查询 SELECT ST\_Segmentize(’LINESTRING(1 2, 3 4)’,0.5);会返回函数名歧义的错误。你需要声明正确的对象类型，例如一个geometry或geography类型的列, 使用函数ST\_GeomFromText, ST\_GeogFromText 或 SELECT ST\_Segmentize(’LINESTRING(1 2, 3 4)’::geometry,0.5);查询

**13.11.4**

**PostGIS 2.0版本新增，实现修改或者有提升的函数**

下面给出的PostGIS函数在PostGIS 2.0版本新添加的，提升的或者在13.11.5 有重大改变的函数

新增的geometry几何类型：在PostGIS 2.0版本引入了 TIN 和 Polyhedral surface类型

**注意**

对于拓扑特性有重大的提升，请参考第 11 章获取更多细节。

**注意**

在PostGIS 2.0版本中，已经集成了栅格类型和栅格功能。这里有很多新的栅格函数列举在这里，并且都是新的函数。请参考第 9章获取更多关于可用栅格函数的细节。在2.0之前的版本有实体表raster\_columns/raster\_overviews ，现在已经变成视图了。像ST\_AddRasterColumn 这样的函数已经移除了，并且用函数AddRasterConstraints，DropRasterConstraints来代替,因此一些创建栅格函数的应用可能需要改变。

**注意**

已经升级的Tiger Geocoder可以支持TIGER 2010 人口普查数据，并且包含在核心的PostGIS 说明文档中，并且已经添加了一个逆编码的函数，请参考12.1 节获取更多信息。

·&&& - 可用版本: 2.0.0。 。如果输入对象A的n-D bounding box与输入对象B的n-D bounding box有交集，则返回TRUE

·<#> -可用版本: 2.0.0 只支持PostgreSQL 9.1+版本的数据库。返回两个几何对象的bounding box之间的距离。该运算符会使用浮点精度（相对于基本的几点对象所使用的double 双精度）。对于其他几何类型，返回的是bounding box的中心点之间的双精度输出距离。在距离排序和使用KNN 功能进行近邻限制输出对象个数时候很有用。（译者注：KNN是机器学习算法之一，被称作近邻算法，输出时候通常会限制输出多少个与输入点类似的对象，可以参考相关书籍 ）

·<-> - 可用版本: 2.0.0 只支持PostgreSQL 9.1+版本的数据库。返回两个点的距离. 对于point或point类型检查，该运算符会使用浮点精度（相对于基本的几点对象所使用的double 双精度）。对于其他几何类型，返回的是bounding box的中心点之间的双精度输出距离。在距离排序和使用KNN 功能进行近邻限制输出对象个数时候很有用。（译者注：KNN是机器学习算法之一，被称作近邻算法，输出时候通常会限制输出多少个与输入点类似的对象，可以参考相关书籍）

· AddEdge - 可用版本: 2.0.0。需要GEOS版本>= 3.3.0。添加一个linestring edge对象到一个edge表中，并且使用指定的linestring几何对象把起点和终点关联到指定topology schema的结点表，然后返回新的（或者已经存在的）edge的edgeid。

· AddFace - 可用版本: 2.0.0。把一个face要素添加到一个topology结构中，然后返回他的id。



PostGIS 2.2.0dev 开发手册

638 / 686

· AddNode - 可用版本: 2.0.0。 添加一个point 结点到一个指定topology schema下的结点表，返回新结点的nodeid。如果point已经存在作为一个结点，那么返回的是已经存在的结点的nodeid。

· AddRasterConstraints - 可用版本: 2.0.0。AddRasterConstraints — 给一个被加载的栅格表的指定列添加一个栅格约束，约束信息包括参考系，缩放比例，块大小，对齐方式，波段，波段类型和一个标志栅格列是否是被锁住的信息。该表必须有约束条件可约束的数据。如果添加约束成功，则返回true，否则提升报错。

· AsGML - 可用版本: 2.0.0。返回一个拓扑结构的GML格式输出。

· CopyTopology - 可用版本: 2.0.0。拷贝一个拓扑的结构(包括结点、边界、面、图层和拓扑结构)。

· DropRasterConstraints - 可用版本: 2.0.0。删除PostGIS的栅格表的列上的栅格约束。如果你需要重装载或更新栅格列的数据，该函数很有用

· Drop\_Indexes\_Generate\_Script - 可用版本: 2.0.0。这个函数会生成一个脚本，用于drop掉所有tiger schema和用户指定的schema下的非primary key和非唯一性索引之外的索引。如果没有指定schema，那么默认的schema是tiger\_data。

· Drop\_State\_Tables\_Generate\_Script - 可用版本: 2.0.0。生成一个SQL脚本，这个脚本可以drop掉一个指定schema下的所有以某个州缩写字母作为前缀的所有表。如果没有指定schema的话，默认的schema是tiger\_data

· Geocode\_Intersection - 可用版本: 2.0.0。输入参数是两条相交的街道以及参数state, city, zip，然后输出两条街道第一个交叉点的可能地址。这些地址都包含一个point几何对象（以NAD83 参考系下的经纬度形式）和一个标准化地址以及一个地址匹配程度的评估值rating，rating值越低表示这个输出地址越匹配输入的地址。输出结果按照rating升序排列（即最匹配的放在前面）。这个函数可以设置一个最大的输出结果记录数，默认是10条记录。

· GetEdgeByPoint - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 根据给定的点，返回和该点相交的边界ID

· GetFaceByPoint - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 根据给定的点，查找与该点相交的面id值。

· GetNodeByPoint - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 找出一个点位置的结点的ID。

· GetNodeEdges - 可用版本: 2.0 。 根据给定的结点，返回连着该结点的有序边。

· GetRingEdges - 可用版本: 2.0 。根据给定的边返回一个环的有序边界集合。

· GetTopologySRID - 可用版本: 2.0.0。根据拓扑名称，返回一个在topology.topology 拓扑表中的拓扑结构的SRID值

· Get\_Tract - 可用版本: 2.0.0。根据geometry几何对象所在的位置，从人口普查表中返回这个位置的人口普查数据。

· Install\_Missing\_Indexes - 可用版本: 2.0.0。找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了，然后添加这些缺失的索引。

· Loader\_Generate\_Census\_Script - 可用版本: 2.0.0。生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据,bg和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。

· Loader\_Generate\_Script - 可用版本: 2.0.0。生成一个指定操作系统平台的shell脚本。这个脚本会下载指定州的Tiger人口普查数据, block groups （bg）和tabblock数据等表，然后加载到tiger\_data这个schema下。每一个州的脚本单独返回一次记录。该函数的最新版本支持Tiger 2010结构化数据，并且会加载到census tract, block groups和 blocks tables表。

· Missing\_Indexes\_Generate\_Script - 可用版本: 2.0.0。找出在地理编码器中进行连接和过滤操作中涉及的表的key列中，哪些索引缺失了。然后输出定义这些索引的SQL DDL语句.

· Polygonize - 可用版本: 2.0.0。找出并注册所有由拓扑边定义的面。

· Reverse\_Geocode - 可用版本: 2.0.0。输入参数是一个已知空间参考系下的point几何对象，返回值是一条包含理论上可能地址的数组和交叉街道的数组。如果参数include\_strnum\_range = true，表示把交叉街道的街道范围也算在内。

· ST\_3DClosestPoint - 可用版本: 2.0.0。返回3维内离g2几何对象最近的g1上的点。这个点也是g1和g2之间最短的线的第一个点

PostGIS 2.2.0dev 开发手册

639 / 686

· ST\_3DDFullyWithin - 可用版本: 2.0.0。如果3D对象的距离在另一个指定的距离值范围内，则返回TRUE

· ST\_3DDWithin - 可用版本: 2.0.0。如果两个几何对象的三维距离指定的距离值范围内，则返回TRUE

· ST\_3DDistance - 可用版本: 2.0.0。对于3维的geometry类型对象，返回两个几何对象的最短笛卡尔距离(基于SRS表)，单位以投影系单位为准。

· ST\_3DIntersects - 可用版本: 2.0.0。如果几何对象在3维空间内相交，则返回TRUE，该函数只适用于Point和LINESTRING类型

· ST\_3DLongestLine - 可用版本: 2.0.0。返回两个几何对象之间最长的3维线段

· ST\_3DMaxDistance - 可用版本: 2.0.0。对于3维的geometry类型对象，返回两个几何对象的最大笛卡尔距离(基于SRS表)，单位以投影系单位为准。

· ST\_3DShortestLine - 可用版本: 2.0.0。返回两个3维几何对象之间最短的线段

· ST\_AddEdgeModFace - 可用版本: 2.0 。在一个face面内添加一条新的边，把这个face面分开，然后修改原始的面并添加一个新的面。

· ST\_AddEdgeNewFaces - 可用版本: 2.0 。在一个face面内添加一条新边界，把这个face面分开，然后删除掉原始的面，并用两个新的面来替换。

· ST\_AsGDALRaster - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。返回指定GDAL栅格格式的栅格瓦片。栅格格式是你的编译库所支持的其中一种。使用ST\_GDALRasters()得到你的库所支持的格式列表。

· ST\_AsJPEG - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。返回指定波段的栅格瓦片为JPEG格式文件。如果没有指定波段，或者指定了但波段为1或超过3，那么只使用第一个波段。若指定了3个波段，那么3个波段都会使用，并且映射到RGB色彩模式中。

· ST\_AsLatLonText - 可用版本: 2.0返回一个给定的度、分、秒表示

· ST\_AsPNG - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。返回指定波段的栅格瓦片为PNG格式文件。如果波段中有1,3或4个波段，但没有指定波段，那么会使用所有波段。如果栅格波段又2个或者多于4个波段，并且没有指定波段，那么只会使用1个波段。波段会被映射到RGB或RGBA模式中。

· ST\_AsRaster - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。把PostGIS 的几何类型对象转换成PostGIS的栅格对象

· ST\_AsTIFF - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。返回栅格指定波段为简单的TIFF图像文件，如果没有指定波段，那么会尝试使用所有的波段。

· ST\_AsX3D - 可用版本: 2.0.0。:需要版本ISO-IEC-19776-1.2-X3DEncodings-XML 以X3D xml节点元素返回返回一个几何对象，格式遵守: ISO-IEC-19776-1.2-X3DEncodings-XML

· ST\_Aspect - 可用版本: 2.0.0。获得栅格表面的坡向 (默认单位是度)。对于地形分析很有用。

· ST\_Band - 可用版本: 2.0.0。根据一个已经存在的栅格，返回一个或更多的波段当做新的栅格。用已有的栅格来创建新的栅格很有用

· ST\_BandIsNoData - 可用版本: 2.0.0。如果波段只设置了NODATA值，则返回true

· ST\_Clip - 可用版本: 2.0.0。返回输入几何对象修改后的栅格。如果波段没有指定，那么会处理所有的波段。如果参数crop没有指定或者值为TRUE，那么输出的栅格会被修改。

· ST\_CollectionHomogenize - 可用版本: 2.0.0。根据给定的geometry collection对象，返回一个最简化的对象

· ST\_ConcaveHull - 可用版本: 2.0.0。返回包含所有输入几何对象的凹包。你可以认为这是一个收缩的包

· ST\_Count - 可用版本: 2.0.0。返回一个栅格给定波段的像素个数。如果没有指定波段，默认取波段1。如果参数exclude\_nodata\_value设置为true，那么这个函数只会统计像素值不为NODATA的像素个数。

PostGIS 2.2.0dev 开发手册

640 / 686

· ST\_CreateTopoGeo - 可用版本: 2.0把一个几何对象的集合添加到一个指定的空拓扑结构中，然后返回处理成功的信息。

· ST\_Distinct4ma - 可用版本: 2.0.0。栅格处理函数，计算近邻栅格单元唯一的像素值的个数。

· ST\_FlipCoordinates - 可用版本: 2.0.0。互换一个输入几何对象的X和Y坐标，对于用户将经度/维度坐标错误输入成维度/经度时候很有用，可以 用这个函数来修复它

· ST\_GDALDrivers - 可用版本: 2.0.0。 - 需要GDAL库>= 1.6.0版本。返回你的GDAL库支持的栅格格式列表。这些格式是你使用函数ST\_AsGDALRaster所输出的类型

· ST\_GeomFromGeoJSON - 可用版本: 2.0.0。 需要 - JSON-C >= 0.9该函数根据一个geojson描述的几何对象，生成一个PostGIS 的geometry对象

· ST\_GetFaceEdges - 可用版本: 2.0返回包围一个面的有序边界集合

· ST\_HasNoBand - 可用版本: 2.0.0。 根据给定的波段，如果没有该波段，则返回true。如果没有指定波段，则默认指的是波段1。

· ST\_HillShade - 可用版本: 2.0.0。根据输入参数azimuth, altitude, brightness和 scale，返回栅格波段的山体阴影。

· ST\_Histogram - 可用版本: 2.0.0。返回概括一个栅格或栅格覆盖数据分布的记录集。如果没有指定参数bins值，默认取autocomputed

· ST\_InterpolatePoint - 可用版本: 2.0.0。根据提供的点，返回输入几何对象离提供的点最近的点的M值

· ST\_IsEmpty - 可用版本: 2.0.0。如果栅格是空的(width = 0 and height = 0)，返回true，否则返回false

· ST\_IsValidDetail - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 返回几何类型是否有效以及无效原因和无效的几何对象是哪些

· ST\_IsValidReason - 可用版本: 2.0 -需要GEOS版本>= 3.3.0返回一个geometry是否是有效的，以及无效的原因

· ST\_MakeLine - 可用版本: 2.0.0。 -根据point或line几何类型创建Linestring类型对象

· ST\_MakeValid - 可用版本: 2.0.0。, 需要GEOS版本-3.3.0。不丢弃顶点的情况下，尝试把无效的几何对象转换成有效的几何对象

· ST\_MapAlgebraExpr - 可用版本: 2.0.0。输入参数是1个波段的函数版本。根据一个用户定义在输入栅格波段和pixeltype值的PostgreSQL代数表达式，生成一个带有一个波段的栅格。如果没有指定波段是哪个，默认取波段1。

· ST\_MapAlgebraExpr - 可用版本: 2.0.0。输入参数是两个栅格波段的版本。根据一个用户定义在两个输入参数栅格波段和pixeltype值的PostgreSQL代数表达式，生成一个带有一个波段的栅格。如果没有指定每一个栅格的波段是哪个，那么默认都取波段1。输出栅格的对其方式 (scale, skew and pixel corners) 和第一个栅格参数相同，边界定义由参数"extenttype"定义，参数"extenttype"的值可以是: INTERSECTION, UNION, FIRST, SECOND。

· ST\_MapAlgebraFct - 可用版本: 2.0.0。输入参数是1个波段的函数版本。根据一个用户定义在输入栅格波段和pixeltype值的PostgreSQL函数，生成一个带有一个波段的栅格。如果没有指定波段是哪个，默认取波段1。

· ST\_MapAlgebraFct - 可用版本: 2.0.0。输入参数是两个栅格波段的版本。根据一个用户定义在两个输入参数栅格波段和pixeltype值的

PostgreSQL函数，生成一个带有一个波段的栅格。如果没有指定每一个栅格的波段是哪个，那么默认都取波段1。

参数"extenttype"如果没有指定值，默认值是: INTERSECTION。

· ST\_MapAlgebraFctNgb - 可用版本: 2.0.0。输入参数是一个波段的函数版本：该函数使用用户自定义的PostgreSQL 函数。该函数会返回一个栅格，其值是一个PLPGSQL自定义函数的结果，该自定义函数与输入栅格波段的近邻值有关。

· ST\_Max4ma - 可用版本: 2.0.0。栅格处理函数，它计算一个邻域的最大像素值。

PostGIS 2.2.0dev 开发手册

641 / 686

· ST\_Mean4ma - 可用版本: 2.0.0。栅格处理函数，它计算一个邻域的平均像素值。

· ST\_Min4ma - 可用版本: 2.0.0。栅格处理函数，它计算一个邻域的最小像素值。

· ST\_ModEdgeHeal - 可用版本: 2.0。解开两个边的连接，通过删除两个边之间的结点，然后更正第一个边的信息（就是第一个边不再带有结

点了），再把第二个边删掉。这个函数返回删除结点的ID。

· ST\_NewEdgeHeal - 可用版本: 2.0 。修复两个边界：先删除连接两个边界的结点，然后删除两个边界，然后再用与第一个边界anedge方向相

同的边界替换这两个边界。

· ST\_Node - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 对LINESTRING对象添加节点进行分隔

· ST\_NumPatches - 可用版本: 2.0.0。返回 Polyhedral Surface的面数.对于非polyhedral 几何类型对象返回NULL值.

· ST\_OffsetCurve - 可用版本: 2.0 - 需要GEOS版本>= 3.2, 在GEOS版本 >= 3.3有提升。根据一个输入的线和偏移的距离，返回一个偏移后的线。对于计算中心线的平行线很有用

· ST\_PatchN - 可用版本: 2.0.0。返回至少包含一个面的geometry对象的第N个面，支持的输入类型是POLYHEDRALSURFACE,

POLYHEDRALSURFACEM. 否则返回NULL值

· ST\_PixelAsPolygon - 可用版本: 2.0.0。返回包含指定行和列数的像素的多边形几何对象。

· ST\_PixelAsPolygons - 可用版本: 2.0.0。返回一个包含一个栅格波段的每个像元的多边形几何对象的记录集，该记录集包含列value，每个像元的X和Y栅格坐标。

· ST\_Project - 可用版本: 2.0.0。返回一个输入类型为距离（单位：米）和方位角（单位：弧度）经投影转换后的Point类型

· ST\_Quantile - 可用版本: 2.0.0。 计算一个栅格或者栅格覆盖的分位点。因此一个像素值可能是一个栅格的25%, 50%, 75% 的分位点。

· ST\_Range4ma - 可用版本: 2.0.0。栅格处理函数，计算邻域的像素值范围。

· ST\_Reclass - 可用版本: 2.0.0。创建一个新的栅格，其波段类型将由将由原始的波段类型转换而来。 如果没有指定nband值，那么默认取波段1。

其他波段都不会改变。使用场景：把一个16BUI的波段转换成8BUI的波段，以便简化导出为可见的格式。

· ST\_RelateMatch - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0. 如果参数intersectionMattrixPattern1 满足了 intersectionMatrixPattern2参数对应的空间关系，则返回TRUE

· ST\_RemEdgeModFace - 可用版本: 2.0。移除一个边，如果移除的边分隔了两个面，删除其中面的其中之一，然后修改另一个，让它占领原本属于两个面的空间，即占有原始的未分开的面。

· ST\_RemEdgeNewFace - 可用版本: 2.0。移除一个边，如果移除的边分隔了两个面，删除原始的面并使用一个新的面进行替换。

· ST\_RemoveRepeatedPoints - 可用版本: 2.0.0。返回一个删除重复点的几何对象

· ST\_Resample - 可用版本: 2.0.0。 需要GDAL库 1.6.1+版本。使用一个指定的重采样算法、新维度参数（width和height）和任意一个网格角点和一个由其他栅格定义的栅

格空间参考属性（由函数的形式参数ref指定）来重采样一个栅格。

· ST\_Rescale - 可用版本: 2.0.0。 需要GDAL库 1.6.1+版本。重采样一个栅格，只调整像素的大小，新的像素值会使用NearestNeighbor (English 或 American 单词拼写方法，

译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采样算法来进行计算。默认

的是NearestNeighbor算法

· ST\_Reskew - 可用版本: 2.0.0。 需要GDAL库 1.6.1+版本。ST\_Reskew —重采样一个栅格，只调整像素的skew(或者说旋转参数)，新的像素值会使用NearestNeighbor (English 或

American 单词拼写方法，译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采

样算法来进行计算。默认的是NearestNeighbor算法

· ST\_SameAlignment - 可用版本: 2.0.0。如果上之间有相同的特征值skew, scale, spatial ref （即SRID），则返回true。如果有不同，则返回false。

· ST\_SetBandIsNoData - 可用版本: 2.0.0。把波段的isnodata标识未TRUE。.

PostGIS 2.2.0dev 开发手册

642 / 686

· ST\_SharedPaths - 可用版本: 2.、0.0。需要GEOS版本>= 3.3.0. 返回两个输入的linestrings/multilinestrings类型对象公用的path路径代表的collection对象

· ST\_Slope - 可用版本: 2.0.0。返回一个栅格波段的坡度（默认以度为单位）。对于地形分析很有用。

· ST\_Snap - 可用版本: 2.0.0。需要GEOS版本>= 3.3.0. 把一个输入的几何对象的片段和顶点切割后重新组装成一个参考类型的几何对象

· ST\_SnapToGrid - 可用版本: 2.0.0。 需要GDAL库 1.6.1+版本。重采样一个栅格，方法是把栅格分割成由任意一个角点像素（gridx & gridy）和可选的像素大小（scalex和scaley）定义的网格。新的像素值会使用NearestNeighbor (English 或 American 单词拼写方法，译者注：英式和美式对同一个单词可能有不同的写法), Bilinear, Cubic, CubicSpline 或Lanczos 重采样算法来进行计算。默认的是NearestNeighbor算法

· ST\_Split - 可用版本: 2.0.0。 Returns a collection of geometries resulting by splitting a geometry.

· ST\_StdDev4ma - 可用版本: 2.0.0。栅格处理函数，计算邻域的像素的标准差。

· ST\_Sum4ma - 可用版本: 2.0.0。 。栅格处理函数，计算一个邻域里面像素值的和。

· ST\_SummaryStats - 可用版本: 2.0.0。 。ST\_SummaryStats —返回一个栅格或者栅格覆盖的一个指定波段的概览统计值，包括：count（像素总数）, sum（像素值之和）, mean（像素平均值）, stddev（像素值标准差）, min（最小像素值）, max（最大像素值）。如果没有指定波段，默认取波段1。

· ST\_Transform - 可用版本: 2.0.0。 ，需要 GDAL 1.6.1+版本。根据给定的投影算法，将一个栅格的一种空间参考系转换到另一个空间参考系中。提供的投影算法有NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos ，默认是NearestNeighbor算法

· ST\_UnaryUnion - 可用版本: 2.0.0。 -需要GEOS版本>= 3.3.0.与函数ST\_Union类似，但处理geometry类型的对象。

· ST\_Union - 可用版本: 2.0.0。 取栅格瓦片的并集生成一个包含1个或更多波段的栅格。

· ST\_ValueCount - 可用版本: 2.0.0。 。ST\_ValueCount —返回一个记录集，包括像素值和指定栅格（或栅格覆盖）的指定波段的像素值在一个值集合内的像素个数。如果没有指定波段，那么默认是波段1。默认也不统计值为NODATA的像素。像素值如果不是整数，那么像素值会进行round四舍五入处理得到一个最接近的整数值。

· TopoElementArray\_Agg - 可用版本: 2.0.0。返回一个topoelementarray数组，它是element\_id，type arrays (topoelements)的集合。

· TopoGeo\_AddLineString -可用版本: 2.0.0。 。把一个linestring几何对象在指定参数tolerance容差下，添加到一个已经存在的拓扑结构中， linestring添加的位置可能会把一个已经存在的edge边界或面分割开。函数会返回edge边界的ID

· TopoGeo\_AddPoint - 可用版本: 2.0.0。把一个点几何对象添加到一个已经存在的拓扑结构中，点添加的位置可能会把一个已经存在的edge边界分割开。

· TopoGeo\_AddPolygon - 可用版本: 2.0.0。 。把一个polygon几何对象在指定参数tolerance容差下，添加到一个已经存在的拓扑结构中，polygon几何对象添加的位置可能会把一个已经存在的edge边界或面分割开。

· TopologySummary - 可用版本: 2.0.0。 根据一个topology name名称，提供一个topology对象里面所有对象类型的概括统计

· Topology\_Load\_Tiger - 可用版本: 2.0.0。Topology\_Load\_Tiger —把tiger数据定义的区域加载到一个PostGIS Topology拓扑结构中，然后把tiger数据转换成拓扑结构所在的空间参考系中以及再把point分割到拓扑结构的tolerance容差精度。

· toTopoGeom - 可用版本: 2.0 把一个简单的几何对象转换成拓扑结构

· ~= - 可用版本: 2.0.0。 如果栅格A的缓冲区和栅格B的缓冲区相同，则返回true。

下面的PostGIS函数是在PostGIS 2.0版本中有提升的：

· AddGeometryColumn - 2.0版本提升：引进了use\_typmod参数，默认使用typmod方式（即用创建一般表字段的方式创建）创建几何类型字段而不是基于约束 (基于约束的方法就是用AddGeometryColumn函数的方式)的方式创建。

· Box2D - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

PostGIS 2.2.0dev 开发手册

643 / 686

· Box3D - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· Geocode - 2.0版本提升：开始支持Tiger 2010 版本结构数据，并且修改了部分处理逻辑来提高速度，解码精确度。新的参数max\_results 对于只想返回最佳结果的使用场景很有用。

· GeometryType - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· Populate\_Geometry\_Columns - 2.0版本提升：引入支持typmod参数，允许控制列的创建类型是typmodiﬁers（类型修改符）还是约束方式。

· ST\_Intersection- 2.0版本提升：引入对栅格空间交集的支持。在2.0.0之前的版本，只支持矢量空间内的对象相交。

· ST\_Intersects - 2.0版本提升：引入支持栅格之间的交集

· ST\_Value - 2.0版本提升：添加了参数exclude\_nodata\_value 支持

· ST\_3DExtent - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Accum - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Afﬁne - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Area - 2.0版本提升： --开始引入对2D polyhedral surface对象的支持

· ST\_AsBinary - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_AsBinary - 2.0版本提升：引入支持更高维度坐标.

· ST\_AsBinary - 2.0版本提升：支持指定geography编码

· ST\_AsEWKB - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_AsEWKT - 2.0版本提升：开始引入支持Geography, Polyhedral surfaces, Triangles 和 TIN

· ST\_AsGML - 2.0版本提升：引入对prefix的支持，引入了对GML3版本的参数option为4的支持，以便允许line对象使用LineString而不是Curve这样的标签。引入了对于Polyhedral surfaces and TINS对象的GML3 的支持。引入了对于参数为option为32时候的支持。

· ST\_AsKML - 2.0版本提升： -添加namespace prefix（默认为kml）的支持，默认没有prefix

· ST\_Azimuth - 2.0版本提升：引入支持geography类型对象.

· ST\_ChangeEdgeGeom - 2.0版本提升：添加了拓扑一致性的强制校验

· ST\_Dimension- 2.0版本提升：支持Polyhedral Surface类型和TIN类型，如果输入参数为空，也不再抛出异常。

· ST\_Dump - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_DumpPoints - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Expand - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Extent - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_ForceRHR - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_Force2D - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_Force3D - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_Force3DZ - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_ForceCollection - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

PostGIS 2.2.0dev 开发手册

644 / 686

· ST\_GMLToSQL - 2.0.0版本提升：开始支持Polyhedral surfaces和 TIN类型对象。

· ST\_GMLToSQL - 2.0版本提升：添加了默认的参数选项

· ST\_GeomFromEWKB - 2.0.0版本提升：开始支持Polyhedral surfaces和 TIN类型对象。

· ST\_GeomFromEWKT - 2.0.0版本提升：开始支持Polyhedral surfaces和 TIN类型对象。

· ST\_GeomFromGML - 2.0.0版本提升：开始支持Polyhedral surfaces和 TIN类型对象。

· ST\_GeomFromGML - 2.0版本提升：添加了默认的参数选项

· ST\_GeometryN - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_GeometryType - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_IsClosed - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_MakeEnvelope - 2.0版本提升：不指定SRID值也能生成一个envelope类型的函数被引入进来

· ST\_MakeValid - 提升了性能，需要版本GEOS-3.3.4支持

· ST\_NPoints - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ST\_NumGeometries - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Relate - 2.0.0版本提升：添加支持指定边界点规则 (需要版本GEOS >= 3.0)

· ST\_Rotate - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Rotate - 2.0.0版本提升，添加了指定旋转的旋转原点的额外参数。

· ST\_RotateX - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_RotateY - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_RotateZ - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Scale - 2.0.0版本提升：开始支持Polyhedral surfaces, Triangles 和 TIN类型对象。

· ST\_Shift\_Longitude - 2.0.0版本提升：开始支持Polyhedral surfaces和 TIN类型对象。

· ST\_Summary - 提升: 2.0.0 版本添加了对geography对象的支持

· ST\_Transform - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

· ValidateTopology - 2.0.0版本提升：更有效的边界交叉检测，修复了之前版本已经明确确定的若干问题。

·&& - 2.0.0版本提升：开始支持Polyhedral surfaces类型对象。

**13.11.5**

**PostGIS 2.0版本实现变更的函数**

下面给出的PostGIS函数在PostGIS2.0版本已经变更了实现，如果做了该版本的升级可能也需要作出相应的改变。

**注意**

大多数不推荐的函数已经移除了。这些函数在1.2版本或者一些没有说明文档的内部函数中已经不再做说明文档了。如果你在使用一个没有说明文档的函数，那么很可能这个函数已经不推荐使用了或者是系统内部的函数，这种函数应当避免使用。如果你的应用或者工具依赖不推荐的函数，请参考 [?qandaentry] 获得更多解决办法的细节。



PostGIS 2.2.0dev 开发手册

645 / 686

**注意**

geometry的bounding box（缓冲区）度量精度已经从float4变更为float8。这会对你使用bounding box运算符和把bounding box转换成geometry对象时候的结果有影响。例如ST\_SetSRID(abbox)这个函数在PostGIS 2.0+中会返回一个比之前版本更精确的结果，这个对接口查询来说会稍微好一些。

**注意**

参数hasnodata 已经被参数exclude\_nodata\_value替换了，这两个参数的意义相同，但后者的字面意思会更加明确。

· AddGeometryColumn - 2.0.0版本更新：视图不能再手动在geometry\_columns中更新信息，然而依赖geometry typmod 方式建的表和没有使用wrapper 函数的视图将会自动注册信息，因为他们继承了他们父表的typmod方式。那些视图使用几何类型函数输出其他的几何类型对象需要被转换成typmod的几何类型对象，以便这些视图中的几何类型的能够在geometry\_columns这个视图中正确注册(就是查询时候能得到这个视图包含的几何类型的记录) · AddGeometryColumn- 2.0.0版本变更：如果你想使用旧的约束方式，使用use\_typmod参数，但默认该值设置为false

· AddGeometryColumn- 2.0.0版本变更:这个函数不再更新geometry\_columns中的记录，因为geometry\_columns现在是从system catalogs(pg\_catalog)读取的数据的视图。该函数默认也不创建约束，相反它使用PostgreSQL内置的类型语句修改方式。因此例如创建一个wgs94 POINT类型的列，等价于这个语句: ALTER TABLEsome\_table ADD COLUMN geom geometry(Point,4326);

· DropGeometryColumn -变更: 2.0.0 版本该函数是提供了后向兼容。既然geometry\_columns是一个依赖于system catalog(译者注：即pg\_catalog)数据信息的视图，因此你可以使用ALTER TABLE这种语法像其他一般类型表的列一样删除一个带有几何类型的列。

· DropGeometryTable - 2.0.0版本变更：该函数是提供了后向兼容。既然geometry\_columns是一个依赖于system catalog(译者注：即pg\_catalog)数据信息的视图，因此你可以使用DROP TABLE这种语法像其他一般类型的表一样删除一个带有几何类型的表。

· Populate\_Geometry\_Columns- 2.0.0版本变更：现在默认使用类型修改符而不是约束条件的方式来约束geometry类型的列，你依然可以通过设置use\_typmod为false的方式来使用老约束条件的方式

· Box3D -2.0.0版本变更：在2.0之前的版本，返回的是box2d几何对象，因为box2d是一个不推荐的类型，因此改成了返回box3d几何对象。

· ST\_ScaleX - 2.0.0版本变更：在WKTRaster 版本中，该函数称为ST\_PixelSizeX.

· ST\_ScaleY - 2.0.0版本变更：在WKTRaster 版本中，该函数称为ST\_PixelSizeY.

· ST\_SetScale - 2.0.0版本变更：在WKTRaster 版本中，该函数称为ST\_SetPixelSize，在2.0.0版本中更新成现在的名字

· ST\_3DExtent - 2.0.0版本变更：在之前的版本这个函数被称之为ST\_Extent3D

· ST\_3DLength - 2.0.0版本变更：在之前的版本这个函数被称之为ST\_Length3D

· ST\_3DLength\_Spheroid - 版本变更: 2.0.0 。在之前的版本中，对于任意非MULTILINESTRING 金额 LINESTRING类型对象，该函数会返回0.在2.0.0版本中该函数对于任意的polygon类型对象，会返回它的周长。

· ST\_3DLength\_Spheroid - 2.0.0版本变更：在之前的版本这个函数被称之为ST\_Length3d\_Spheroid

· ST\_3DMakeBox - 2.0.0版本变更：在之前的版本这个函数被称之为ST\_MakeBox3D

· ST\_3DPerimeter - 2.0.0版本变更：在之前的版本这个函数被称之为ST\_Perimeter3D

· ST\_AsBinary - 2.0.0版本变更：该函数的输入不能是未知的，必须是一个几何对象。如下查询SELECT ST\_AsBinary(’POINT(1 2)’) 不再支持，你会得到一个st\_asbinary(unknown) 不唯一的错误.应该把代码修改为ST\_AsBinary(’POINT(1 2)’::geometry);. 如果该查询不可用，那么需要执行代码文件legacy.sql.（译者注：使用psql执行）



PostGIS 2.2.0dev 开发手册

646 / 686

· ST\_AsGML - 2.0.0版本变更：开始使用默认参数

· ST\_AsGeoJSON - 2.0.0版本变更：开始使用默认参数，并支持命名参数

· ST\_AsKML - 2.0.0版本变更： -开始使用默认参数，并支持命名参数

· ST\_AsSVG - 2.0.0版本变更：开始使用默认参数，并支持命名参数

· ST\_EndPoint- 2.0.0版本变更：不再对单MULTILINESTRING类型支持。在旧版本的PostGIS中，该函数能够很好的支持单

MULTILINESTRING类型，并返回起始点，从2.0.0开始，该函数会返回NULL值。旧的方式是不规范的特性，但对于那些把数据

存储为MULTILINESTRING的用户来说，在2.0.0版本的PostGIS将会返回NULL值

· ST\_GeomFromText - 2.0.0版本变更：在早先的PostGIS版本，ST\_GeomFromText(‘GEOMETRYCOLLECTION(EMPTY)’) 允许执行，而在PostGIS 2.0.0 版本为了遵守SQL/MM规范，这个写法是非法的。现在应该这样写

ST\_GeomFromText(‘GEOMETRYCOLLECTION EMPTY’)

· ST\_GeometryN - 2.0.0版本变更： 2.0.0之前的版本对于单几何类型对象会返回NULL值，从2.0.0版本起，对于ST\_GeometryN(..,1) 这样的情况将会返回geometry类型对象

· ST\_IsEmpty - 2.0.0版本变更：在之前的PostGIS版本，ST\_GeomFromText(’GEOMETRYCOLLECTION(EMPTY)’) 是允许的，在PostGIS 2.0.0版本及之后，为了更好的遵守SQL/MM规范，这种写法是禁止的，不符合语法的

· ST\_Length - 2.0.0版本变更：该版本重大突破–先前版本该函数对于MULTI/POLYGON类型的geography对象，该函数会返回POLYGON/MULTIPOLYGON类型对象的周长。在2.0.0版本，该函数会返回0，以便于geometry类型对象保持一致。如果你想计算一个polygon类型对象的周长，使用函数 ST\_Perimete

· ST\_LocateAlong - 2.0.0版本变更：该函数之前的版本叫做ST\_Locate\_Along\_Measure，老的函数名不推荐使用，将来也不会在支持

· ST\_LocateBetween - 2.0.0版本变更：在之前的版本，该函数被称作ST\_Locate\_Between\_Measures，旧的函数名不推荐使用，将来也会移除，但当前为了保持后向兼容，依然可用

· ST\_ModEdgeSplit - 2.0版本变更： -在之前的版本，这个函数名被误定义为ST\_ModEdgesSplit

· ST\_NumGeometries - 2.0.0版本变更：2.0.0版本之前的版本，如果geometry类型不是collection或/MULTI类型，该函数将会返回NULL值，在2.0.0+版本，对于单个几何对象会返回1，比如POLYGON, LINESTRING, POINT类型

· ST\_PointN - 2.0.0版本变更：该函数不再支持单multilinestring几何对象，在早期的版本，该函数可以很好的支持但line

multilinestring类型，并返回该几何对象的起点。在2.0.0版本，对于其他multistring类型都返回NULL

· ST\_StartPoint- 2.0.0版本变更：不再对单MULTILINESTRING类型支持。在旧版本的PostGIS中，该函数能够很好的支持单MULTILINESTRING类型，并返回起始点，从2.0.0开始，该函数会返回NULL值。旧的方式是不规范的特性，但对于那些把数据存储为MULTILINESTRING的用户来说，在2.0.0版本的PostGIS将会返回NULL值。

**13.11.6**

**PostGIS 1.5版本新增的，实现修改的或者提升的函数**

下面给出的PostGIS 函数是在1.5版本引入或者有所提升的函数

· PostGIS\_LibXML\_Version - 可用版本: 1.5 。返回 libxml2 库的版本号.

· ST\_AddMeasure- 可用版本: 1.5.0 ST\_AddMeasure —在输入几何对象的起点和终点之间进行线性插值，返回一个新的带有M值的几何对象，如果输入的几何对象没有测量维度的M值，那么就自动驾驶。如果输入的几何对象有M值，那么会覆盖掉。目前该函数只支持LINESTRINGS和 MULTILINESTRINGS。

· ST\_AsBinary - 可用版本: 1.5.0 引入支持geography类型对象。返回一个没有SRID信息的WKB描述的geometry或geography对象.

· ST\_AsGML - 可用版本: 1.5.0 引入支持geography类型对象。返回GML 2或GML 3版本格式描述的几何对象

.

PostGIS 2.2.0dev 开发手册

647 / 686

·

ST\_AsGeoJSON - 可用版本: 1.5.0 引入支持geography类型对象。返回一个GeoJSON格式（译者注：GeoJSON是一种类似json的格式数据，适用于地理编码）的几何对象.

· ST\_AsText - 可用版本: 1.5 - 引入支持geography类型对象。返回不含有SRID信息的geometry或geography对象的WKT表示

· ST\_Buffer -可用版本: 1.5 –这次2.2.0版本ST\_Buffer提升了对不同的end cap 和join类型（译者注：这两个专业术语应该是不同几何对象的结合方式，这里给出原文）的支持。在如下场景很有用，例如把LINESTRING类型对象转换成polygon类型对象， LINESTRING之间结合方式是直角而不是圆角。对于geography类型对象的简单函数包装也支持。-需要版本 GEOS >= 3.2 来充分利用geometry功能。对geometry类型对象来说，返回以给定点为中心点，距离小于或等于指定距离的所有点组成的几何对象。计算方式是按照spatial\_ref\_sys表中对该类型的几何对象的描述来计算的。对于geography类型对象：使用一个2D维度转换包装器。PostGIS 1.5版本引入了对不同的end cap和mitre参数的设置来控制几何对象的形状。buffer\_style参数控制如下：quad\_segs=#,endcap=round|ﬂat|square,join=round|mitre|bevel,mitre\_limit=#.#

· ST\_ClosestPoint - 可用版本: 1.5.0 。返回2维平面内离g2几何对象最近的g1上的点。这个点也是g1和g2之间最短的线的第一个点

· ST\_CollectionExtract - 可用版本: 1.5.0 。根据输入的(multi)几何对象，返回一个(multi)几何对象，返回的几何对象只包含指定的类型的元素

· ST\_Covers - 可用版本: 1.5 - 引入支持geography类型对象。如果geometry或geography对象B的所有点都不在geometry或geography对象A的外部，则返回1（即TRUE）。

· ST\_DFullyWithin - 可用版本: 1.5.0。如果输入的所有几何对象之间的距离都在一个指定的距离内，那么返回TRUE

· ST\_DWithin- 可用版本: 1.5.0 引入支持geography类型对象。引入支持geography对象。如果输入的几何对象的都在以一个点为中心的指定的距离范围内则返回TRUE。对于geometry类型对象，单位以SRS表中的距离单位为标准。如果是geography类型，单位是米。默认的计算方式是use\_spheroid=true (以椭球面为参考系), 如果想更快的检索出所需要的点，使用use\_spheroid=false，该参数表示以正规的球体为参考系。.

· ST\_Distance - 可用版本: 1.5.0 1.5版本引入支持geography对象. 提升了2维处理能力以便处理更大更多的顶点几何对象。对于geometry类型对象，返回两个几何对象的2维的最小笛卡尔距离。对于geography类型对象，返回WGS84参考系两个geography对象之间的最小空间距离，距离单位是米

· ST\_Distance\_Sphere -可用版本: 1.5 -支持除了点类型以外的几何对象，早前的版本只支持点类型对象。返回两个以经纬度表示的几何对象的最小距离。该函数使用一个半径为6370986 球体做参照。该函数的计算比ST\_Distance\_Spheroid要快，但是精度要差一些（译者注：因为地球不能完全看做一个正规的球体）。1.5版本之前的PostGIS的该函数只实现了对Point类型对象的支持

· ST\_Distance\_Spheroid -可用版本: 1.5 -支持除了点类型以外的几何对象，早前的版本只支持点类型对象。根据给定的椭球面参考系，计算两个经纬度表示的几何对象的最小距离。PostGIS 1.5版本之前只支持Point类型对象

· ST\_DumpPoints- 可用版本: 1.5.0。返回组成一个几何对象的所有点。

· ST\_Envelope -该函数可用版本: 1.5.0 版本的输出的精度是float8，而不是float4。从提供的geometry类型中返回一个box的边界值精度为float8的geometry类型（译者注：是一个凸包）

· ST\_GMLToSQL - 可用版本: 1.5, requires libxml2 1.6+。根据GML表述的对象返回一个ST\_Geometry值，这个函数是ST\_GeomFromGML的别名。

· ST\_GeomFromGML - 可用版本: 1.5, requires libxml2 1.6+ 。根据一个geometry对象的GML描述，返回一个PostGIS的geometry类型对象

· ST\_GeomFromKML - 可用版本: 1.5,libxml2 2.6+ 。该函数根据一个KML描述的几何对象，生成一个PostGIS 的geometry对象

· ~= - 可用版本: 1.5.0修改了实现方式。如果A的bounding box和B的bounding box相同，则返回TRUE

· ST\_HausdorffDistance - 可用版本: 1.5.0 - 需要GEO、S版本>= 3.2.0。ST\_HausdorffDistance —返回两个几何对象的Hausdorff 距离。该距离用来表示两个几何对象有多类似或有多不相同。距离单位是这两个几何对象所在的空间参考系所用的距离单位

PostGIS 2.2.0dev 开发手册

648 / 686

· ST\_Intersection - 可用版本: 1.5 引入支持geography类型对象。返回几何对象相交的点集对应的geometry对象。对于geography类型对象，该函数会先将其转化成geometry类型对象，然后再转换成WGS84下的坐标

· ST\_Intersects - 可用版本: 1.5 引入支持geography类型对象。如果Geometries/Geography在2维空间内有相交（有共同的空间部分），则返回TRUE。如果他们不相交，那么

返回FALSE。对于geography类型对象—误差是0.00001米（因此许多很靠近的点会被认为是相交（即重合））

· ST\_Length - 可用版本: 1.5.0 。返回一个LINESTRING或MULTILINESTRING类型对象的2维长度。长度单位对于geometry类型对象来说以空间参考系规定的单位为标准，geography类型以椭圆参考系为参考系，单位是米。

· ST\_LongestLine - 可用版本: 1.5.0 。返回两个几何对象之间的2维最长距离。如果有多个线段都是最长线段，那么返回第一个最长的。该线段的起点在g1，终点在g2.该函数返回的线段的长度与用函数ST\_MaxDistance计算g1和g2的距离值一样。

· ST\_MakeEnvelope - 可用版本: 1.5。根据给定的最小值范围和最大值范围生成一个矩形，输入值必须是SRS（spatial\_reference\_system表）规定的SRID值

· ST\_MaxDistance - 可用版本: 1.5.0 。返回两个几何对象的最长距离，距离单位是投影单位

· ST\_ShortestLine - 可用版本: 1.5.0 。返回两个几何对象之间最短的2维线段

·&& -可用版本: 1.5.0引入支持geography对象.。如果输入对象A的2D bounding box与输入对象B的2D bounding box有交集，则返回TRUE

**13.11.7**

**PostGIS 1.4版本引入的新的，实现方式改变了的或有提升的函数**

下面的给出的PostGIS 函数是在1.4版本引入或者有提升的函数：

·Populate\_Geometry\_Columns -确保geometry类型的列是用typmod方式来定义的或者有合适的空间约束条件，这样才能保证

这些列能被正确地注册到geometry\_columns这个视图中。默认情况下会将所有的非typmod定义的geometry类型列转换为

typmod方式的，如果想用老的方式，需要把use\_typmod值设为false。可用版本: 1.4.0

· ST\_AsSVG -根据给定geometry或geography对象，在一个SVG 数据中返回一个geometry对象。可用版本: 1.4.0 从1.4.0版本开始，在绝对路径中添加了L命令来保持和SVG规范一致，参考http://www.w3.org/TR/SVG/paths.html#PathDataBNF

· ST\_Collect-从其他几何类型对象的collection 返回一个具体的ST\_Geometry值（对应的对象）。可用版本: 1.4.0 - 引入函数ST\_Collect(geomarray)支持. ST\_Collect函数用于更快地处理更多的几何对象

· ST\_ContainsProperly -如果输入几何对象B和几何对象A的内部相交，但不和A的边界（或外部）有接触，那么返回TRUE。

可用版本: 1.4.0 - 需要GEOS版本>= 3.1.0.

· ST\_Extent -一个返回包含多个geometry对象的bounding box的聚集函数。可用版本: 1.4.0

· ST\_GeoHash -返回一个几何对象的GeoHash表示。可用版本: 1.4.0

· ST\_IsValidReason -返回一个geometry是否是有效的，以及无效的原因可用版本: 1.4 –需要GEOS版本>= 3.1.0.

· ST\_LineCrossingDirection -根据两个LINESTRING，返回一个范围在-3到3之间的数字，该数字表示这两个LINESTRING的相

交方式。0表示这两个几何对象没有相交关系可用版本: 1.4

· ST\_LocateBetweenElevations -返回高度值在指定范围内的几何对象（collection）。目前只支持 3D, 4D LINESTRINGS 和MULTILINESTRINGS可用版本: 1.4.0

· ST\_MakeLine -根据point或line几何类型创建Linestring类型对象。可用版本: 1.4.0 - ST\_MakeLine(geomarray) 函数被引入，ST\_MakeLine 聚集函数性能提升，处理更多点会更快

· ST\_MinimumBoundingCircle -返回包含一个几何对象的最小的Circle Polygon对象。默认对四分之一圆做48等分弧段分割。可用版本: 1.4.0 - 需要GEOS版本

· ST\_Union -返回一个代表一些几何对象的并集的几何对象。可用版本: 1.4.0 - ST\_Union性能有所提升。引入了ST\_Union(geomarray) 支持，在PostgreSQL中运行得也更快。如果你使用 GEOS 3.1.0+版本，函数 ST\_Union 会使用更快的Cascaded Union 算法，该算法介绍如下http://blog.cleverelephant.ca/2009/01/must-faster-unions-in-postgis-14.html。

PostGIS 2.2.0dev 开发手册

649 / 686

**13.11.8**

**PostGIS 1.3版本新增函数**

下面是PostGIS 1.3版本引入的新函数

· ST\_AsGML -返回GML 2或GML 3版本格式描述的几何对象。可用版本: 1.3.2

· ST\_AsGeoJSON -返回一个GeoJSON格式（译者注：GeoJSON是一种类似json的格式数据，适用于地理编码）的几何对象。可用版本: 1.3.4

· ST\_SimplifyPreserveTopology -使用Douglas-Peucker算法，根据给定的几何对象，返回一个简化版的几何对象。该函数会避免创建新的派生的无效的几何对象（特别是多边形）。可用版本: 1.3.3