26 / 686

**Chapter 3**

**PostGIS 常见问题**

1. 我在哪里可以找到关于PostGIS的相关教程、指导及深入探讨呢？

OpenGeo提供了一步一步从入门到精通的课程Introduction to PostGIS。它提供了包括数据和入门指导的OpenGeo套件。可能这是最好的PostGIS教程。BostonGIS 也有一个傻瓜式入门指导 [PostGIS almost idiot’s](http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01)

[guide on getting started.](http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01)后者更偏向于面向Windows用户

2. 我的应用和桌面工具都支持PostGIS 1.5版本，但是不支持PostGIS 2.0版本，我该怎样解决这个问题？

在PostGIS 2.0版本，移除了大量不推荐的函数。这影响了一些应用，举个例子来说，影响包括Geoserver, MapServer, QuantumGIS和 OpenJump。有许多办法来解决这个问题。对于第三方工具应用，你可以升级到已经修复了这些不支持问题的最新版本。对于你自己的代码，你可以移除代码里面不推荐函数的使用。大部分函数都是ST\_Union, ST\_Length函数等的别名函数，万不得已的情况下，你可以安装整个legacy.sql 脚本或者你需要的部分legacy.sql 脚本。脚本文件legacy.sql 放在和postgis.sql文件相同的目录下。安装完postgis.sql 和 spatial\_ref\_sys.sql后，你可以安装这个文件来恢复大概200个左右被移除的旧函数。

3. 当我使用osm2pgsql命令加载OpenStreetMap数据时候，得到一个失败错误：: ERROR: operator class "gist\_geometry\_ops" does not exist for access method "gist" Error occurred，而在PostGIS 1.5版运行得很好

答：在PostGIS 2版本，默认的geometry 算子gist\_geometry\_ops更新为gist\_geometry\_ops\_2d，而gist\_geometry\_ops已经完全被移除了。做出这样改变的原因是PostGIS 2 引入了对3D对象的Nd 空间索引的支持，同时旧的算子名称是混乱且错误的。

一些旧的应用比如创建了表和索引，会引用算子的名称。如果你需要的是默认的2D 索引，那么这种引用是不必要的。因此你可以把以前不好的创建索引方式：

CREATE INDEX idx\_my\_table\_geom ON my\_table USING gist(geom gist\_geometry\_ops);

改成好的

CREATE INDEX idx\_my\_table\_geom ON my\_table USING gist(geom);

唯一需要注意的地方是你需要指定算子的地方，比如你想向下面这样创建3D空间索引：

CREATE INDEX idx\_my\_super3d\_geom ON my\_super3d USING gist(geom gist\_geometry\_ops\_nd);

如果你不信碰到那些你不能改变的带有老的算子的 gist\_geometry\_ops代码，那么你可以使用打包在PostGIS 2.0.2+版本中的脚本legacy\_gist.sql来创建老的算子。然而如果你使用这个修复办法，建议你稍后删除老的索引，然后重新创建不带参数gist\_geometry\_ops的索引。这会在你在你将来再次升级的时候给你减少痛苦。

4. 我正在运行PostgreSQL 9.0版本，我却再也不能通过OpenJump, Safe FME, 和其他工具查看读取几何类型数据了，怎么办？

答：在PostgreSQL 9.0+版本中，默认的对于bytea数据的编码方式已经改成hex编码，而老的JDBC驱动依然假定进行字符转义。这会影响到一些使用老的JDBC驱动的Java应用或者使用npgsql驱动的.NET应用，而这些驱动需要ST\_AsBinary老的输出结果。对于该问题有两种解决办法。你可以从http://jdbc.postgresql.org/download.html 上面获取最新的PostgreSQL 9.0版本的驱动，然后升级你的JDBC驱动。

27 / 686

如果你使用Npgsql 2.0.11或者更高版本，那么你可以从http://pgfoundry.org/frs/?group\_id=1000140 下载最新驱动，如 [Francisco Figueiredo’s NpgSQL2.0.11 released blog entry](http://fxjr.blogspot.com/2010/11/npgsql-2011-released.html)版本博客所述。如果你不需要升级你的PostgreSQL驱动，那么你可以通过下面的语句将bytea\_output输出格式改成旧的方式：

ALTER DATABASE mypostgisdb SET bytea\_output=‘escape’;

5.我试着使用PgAdmin来查看我的几何数据类型列，但却是空的，怎么回事呢？

答：PgAdmin 对于大型几何数据类型不展示任何东西。最好在办法查看你的几何类型数据是：

-- this should return no records if all your geom fields are filled in

SELECT somefield FROM mytable WHERE geom IS NULL;

-- To tell just how large your geometry is do a query of the form

--which will tell you the most number of points you have in any of your geometry

←

columns

SELECT MAX(ST\_NPoints(geom)) FROM sometable;

6. 用PostGIS，我可以存储哪些类型的几何对象？

你可以在PostGIS中存储如下类型的几何对象point, line, polygon, multipoint, multiline, multipolygon, 和 geometry collections。在PostGIS 2.0及更高版本，你还可以以基础几何类型方式存储TINS 和 Polyhedral Surface 对象。这些几何对象都是用Open GIS Well Known Text Format (带有 XYZ,XYM,XYZM 这样的格式扩展)表示的。目前PostGIS支持三大数据类型：标准的OGC 几何数据类型，该数据类型坐标使用平面坐标系度量，地理数据类型，该数据类型使用大地坐标系（非OGC标准，但你可以在Microsoft SQL Server 2008+找到类似的数据类型），只有WGS 84 参考系的经纬度（SRID:4326）坐标是地理数据类型支持的。PostGIS最新的空间数据类型成员是栅格类型，用于存储和分析栅格数据。栅格有自己的FAQ，参考第10和第9章获取更多信息。

7. 我有点迷糊，我该使用哪种数据类型来存储数据呢？几何类型还是地理类型呢？

答：简单来讲：geography是一个新的数据类型，它支持大范围的距离测量计算，但是在当前该类型上的大部分计算要比在geometry类型上慢得多。如果你使用geography类型-- 你不必学习太多关于平面参考系的知识。如果你只关心距离和长度的计算，并且计算的数据对象是全世界范围内的数据，那么geography类型是最好的类型。geometry数据类型是一个更古老的数据类型，它有很多函数支持，很多第三方工具的支持，在该类型上的操作运算通常也会更快——有时候对于大型几何类型对象有着10倍的速度提升。如果你比较习惯空间参考系或者你处理的是局部数据，即数据都是处于单一的参考系中（SRID值一样），或者你需要许多空间任务处理，那么geometry数据类型是最好的选择。注意：这两种数据类型转换都很简单，因而可以获取到每一种类型的优点。参考13.10 章节查看当前支持哪些类型哪些不支持。复杂地说：参考我们在4.2.2节里面的详细讨论和function type matrix（函数类型矩阵）。

8.我对地理类型有更强烈的问题，比如填满一个地理类型列需要多大的地理面积数据呢？目前我得到了较合理的回答。对于极点类型，字段里面的所有东西必须要适配半球面（像SQL Server 2008那样）以及速度等等吗？

答：你的问题太深入复杂了（译者注：我也觉得这是个什么操蛋的问题！）,以致于在这里无法完整地回答，参考4.2.3节

9. 我该怎样把GIS对象插入到数据库中？

答：首先你要创建一个带有geometry或者geography类型字段的表来保存GIS数据。存储geography类型数据与geometry类型稍微有一些不同。参考4.2.1 获取关于geography类型数据存储的细节。对于geometry类型，使用psql连接数据库然后执行下面的SQL:

CREATE TABLE gtest ( gid serial primary key, name varchar(20) , geom geometry(LINESTRING) );

如果geometry类型列定义失败，你可能没有把PostGIS函数和对象加载到数据库中，或者使用了2.0版本之前的PostGIS数据库。参考2.4节。然后你就可以使用SQL插入语句来讲geometry类型数据插入到数据库中。GIS对象本身是用OpenGIS 委员会的WKT格式描述的。

28 / 686

INSERT INTO gtest (ID, NAME, GEOM)

VALUES (

1,

‘First Geometry’,

ST\_GeomFromText(‘LINESTRING(2 3,4 5,6 5,7 8)’)

);

关于GIS对象的更多信息，请参考object reference。若想查看表中的GIS数据：

SELECT id, name, ST\_AsText(geom) AS geom FROM gtest;

The return value should look something like this:

id | name

| geom

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

1 | First Geometry | LINESTRING(2 3,4 5,6 5,7 8)

(1 row)

10. 我该怎样构建一个空间查询?

就像你构建其他数据库的任何SQL查询方式一样，可以返回正常的值、函数、逻辑值等。对于空间查询，有两个特别重要的场景需要需要在查询时候记住：是否有空间索引你可以使用；你是否在大的几何对象上做耗费资源的计算。一般情况下，你应该要使用交集运算符（&&）来做交集运算，该运算符可以通过构建一个bounding box来缩小范围。&&运算符好用的理由是如果该列有一个可用的索引来加速查询，&&可以充分利用。这样就可以大大加快查询。你应当也充分利用一些空间函数，例如ST\_Distance(),ST\_Intersects(), ST\_Contains() and ST\_Within()等等函数来缩小你的搜寻范围。大部分空间查询应该要考虑先进行查询时候的索引测试和函数选用测试。索引测试有助于将返回的元祖限制到只满足我们感兴趣的条件范围内。空间函数用于测试给定的条件是否符合要求。

SELECT id, the\_geom

FROM thetable

WHERE ST\_Contains(the\_geom,’POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))’);

11. 我怎样提高在大表上的空间查询速度呢?

对于支持事务的空间数据库中的大型表的快速查询来说，有一个好的索引是很重要的。使用CREATE INDEX语句来在一个表的几何数据类型列上面创建空间索引：

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

"USING GIST" 参数告诉服务器要使用GiST（Generalized Search Tree ）索引

**注意**

GiST 索引被假定是无损的。无损索引使用代理对象（例如在空间对象中，使用bounding box当做代理对象）来创建索引

查询时候你应该保证PostgreSQL的查询优化器搜集到足够的关于你的索引信息，以便让查询优化器在何时使用索引做出正确的决策。为此，你必须搜集几何数据表的统计信息。对于PostgreSQL

8.0.x 及更高版本，请执行VACUUM ANALYZE命令搜集统计信息，对于PostgreSQL 7.4.x 及以下版本，执行SELECT UPDATE\_GEOMETRY\_STATS() 命令搜集统计信息。

12. 为什么PostGIS不支持R-Tree索引呢？

早期版本的PostGIS使用PostgreSQL 的R-Tree 索引。然而，从PostGIS 0.6版本开始，PostGIS已经完全抛弃使用PostgreSQL 的R-Tree索引了，并提供基于PostgreSQL R-Tree索引的GiST索引。我们的测试显示，GiST索引的性能已经可以和原生的R-Tree索引搜索性能不相上下。原生的PostgreSQL R-Tree索引有两个限制，导致R-Tree索引已经不适合GIS特性了（注意：这些限制是目前PostgreSQL版本的原生的R-Tree索引实现引起的，而不是一般意义上的R-Tree索引，比如其他数据库对于R-Tree索引实现可能不会有这些限制）：



29 / 686

·在PostgreSQL中R-Tree 索引不能处理超过8K字节的大型对象，而GiST 索引可以使用bounding box取代索引列对象本身这样的无损处理技巧来处理。

·在PostgreSQL中，R-Tree 索引是NULL值不安全的，因此在一个包含null值的几何数据类型列上面创建R-Tree索引会失败

13. 为什么我必须使用函数AddGeometryColumn() 和其他OpenGIS规范的一些东西？

如果你不想使用OpenGIS支持的函数，你不必使用。可以像老版本那样使用CREATE语句来创建几何类型数据列。你的几何对象的SRID值都会是-1，并且OpenGIS的元数据表将没有该字段的正确信息。然而这会让你基于PostGIS的大部分应用都会失效，一般情况下还是建议你使用函数AddGeometryColumn()来创建几何数据类型列。MapServer 是一个可以充分利用geometry\_columns 元数据信息的GIS服务器，MapServer 可以使用几何数据列的SRID值在正确的投影图上做几何对象的特性的临时性投影

14. 半径范围搜索的最好办法是什么？

答：为了最高效地使用数据库，半径搜索最好要与bounding box测试一起使用。bounding box 测试使用空间索引，会获取半径范围内的所有数据子集。函数ST\_DWithin(geometry, geometry, distance) 是一个很好用的使用索引的半径搜索函数。它通过创建一个足够大的包含半径距离的矩形，然后索引结果范围内再进行精确的距离匹配搜索。例如找到POINT(1000 1000) 100米范围内的查询如下：

SELECT \* FROM geotable

WHERE ST\_DWithin(geocolumn, ‘POINT(1000 1000)’, 100.0);

15. 我该怎样在查询中做坐标投影转换呢？

答：要想做一个坐标投影转换，源坐标系和目的参考系都必须是定义在SPATIAL\_REF\_SYS表中的参考系，被转换的几何列必须依据有SRID志。一旦这些要求具备了，投影转换就是寻找目标SRID值这样简单的事情。下面是将一个几何对象投影成NAD 83经纬度坐标系。下面的查询只有在列the\_geom的SRID值不是-1（就是没有定义SRID值）

SELECT ST\_Transform(the\_geom,4269) FROM geotable;

16.我在我的一个相当大的几何字段上面使用函数ST\_AsEWKT 和函数 ST\_AsText 时，返回的值却是空。这是怎么回事呢？

答：你可能使用了PgAdmin或者一些其他不支持大型text文本输出的工具。如果的几何对象真的很大，在这些工具里面会显示成空白。如果你真想看它的话，使用PSQL或者以WKT格式输出。

--检查几何对象是真正的空值的数量：

SELECT count(gid) FROM geotable WHERE the\_geom IS NULL;

17. 当我在取两个几何对象的交集时候，我明明知道这两个几何对象绝逼可以相交，为毛这个函数却返回这两个几何对象不可以相交呢，出了什么幺蛾子？

答：一般情况下，这种情况经常发生。可能原因是你的几何对象是无效的。使用感受ST\_IsValid来检测你假定可以相交的对象是否有效，因为你不知道的情况下，函数ST\_AsText会丢掉一些小数位，导致精度损失

18.我正在发布使用PostGIS的软件，这是否意味着我的软件必须要像PostGIS一样，使用GPL规范来授权呢？

如果我使用了PostGIS，我必须要公开我的软件代码吗？

答：几乎可以确定不用。比如说，Oracle数据库是运行在Linux上面的，Linux是GPL规范授权的，但Oracle不是，难道运行在Linux上的Oracle必须使用GPL规范来分发（售卖）产品吗？当然不是。你的软件可以在你想的任何授权下使用PostgreSQL/PostGIS。唯一的例外是，你修改了PostGIS源码并发布了你修改过的PostGIS版本。在这种情况下，你必须共享出你修改过的PostGIS代码（不是运行于PostGIS上面的应用的代码）。即便在这样有限制的情况下，你也只是必须向那些你分发二进制软件包的人分发源码。GPL规范不要求你公布自己的源码，只要求你向那些你分发二进制软件包的人分发源码