北京市地图和道路匹配情况

北京市地图

Sharpfile文件

目前，北京市地图sharpfile文件里面地图数据比较简单，数据比较旧。

北京市sharpfile地图文件说明

地图共有道路路段10594条，其中，包含名字路段有3282条，没有名字路段有7312条。

Sharpfile文件有固定的读取格式，详情请参考sharpfile white paper链接： http://wenku.baidu.com/link?url=RVaziti-yt7gDqe0VLOiq1ptyQ90EQUD726wDXkrE2km6X4WUmHrVOxnfyiRGm8hcTdB7qmsN6YvRshfOX-WgPcITecbgny9sEh-qeHEDre

道路概况：

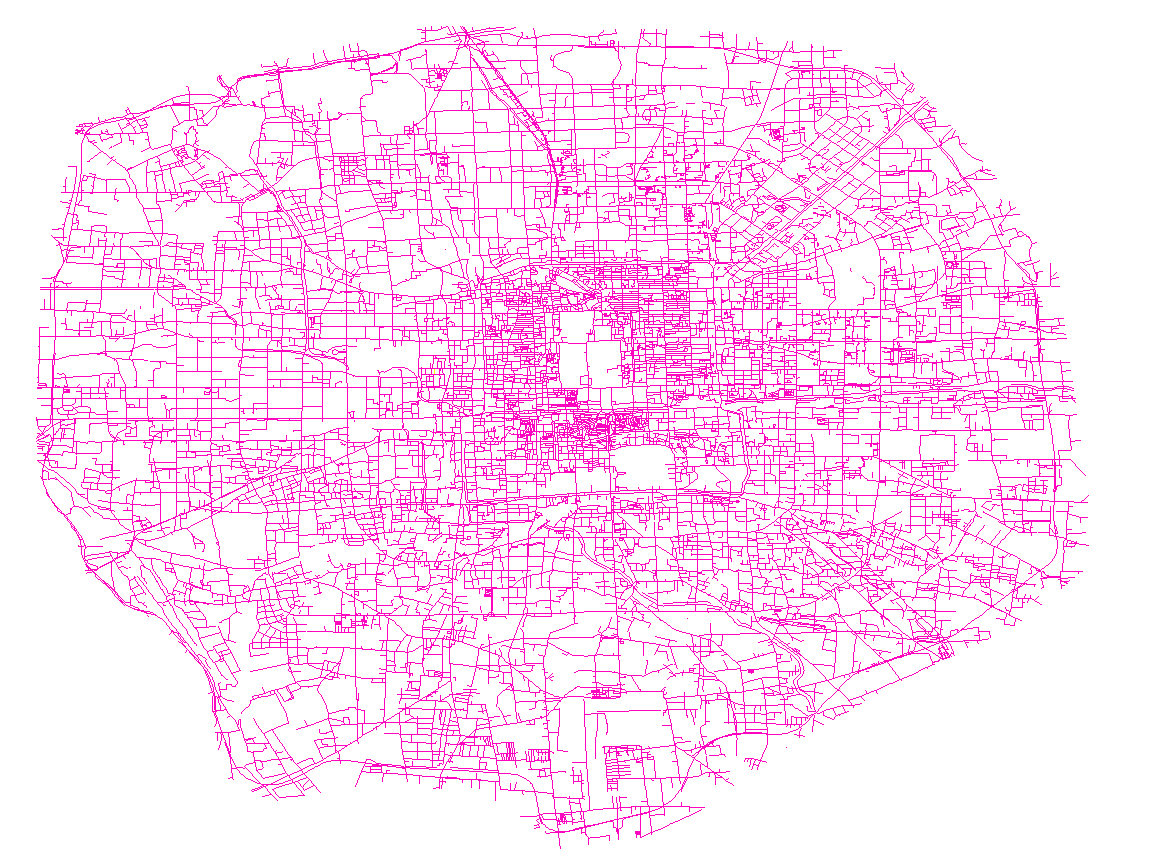


图1 北京市道路网络地图

北京市道路网络图范围信息如下：最小经度为116.199814，最大经度为116.554159，最小纬度为39.750665，最大纬度为40.027734。基本覆盖五环内道路。

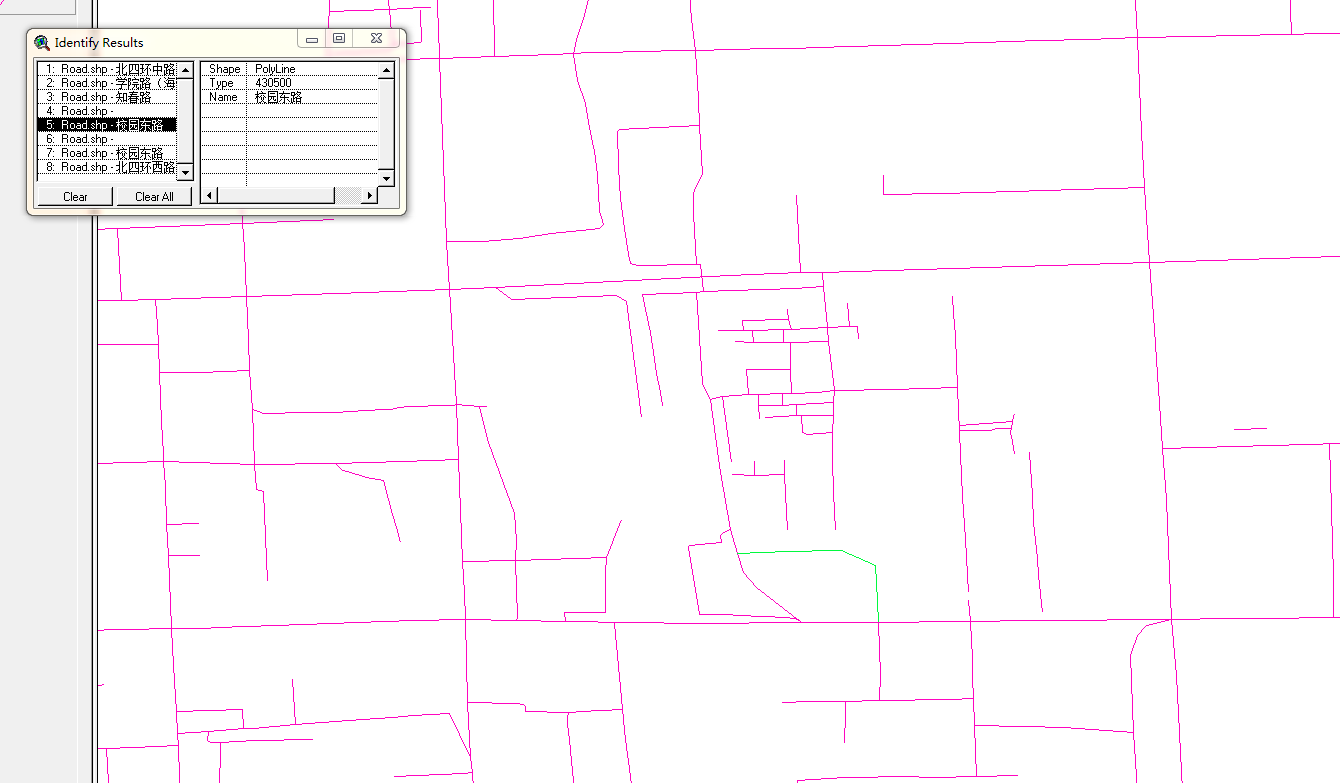


图2 北航附近道路网络图

绿色路段为大运村一侧路段，可以看到道路网络比较简单，上面水平方向长线段为北四环中路，其用一条直线表现出道路，下面水平方向长线段为知春路。

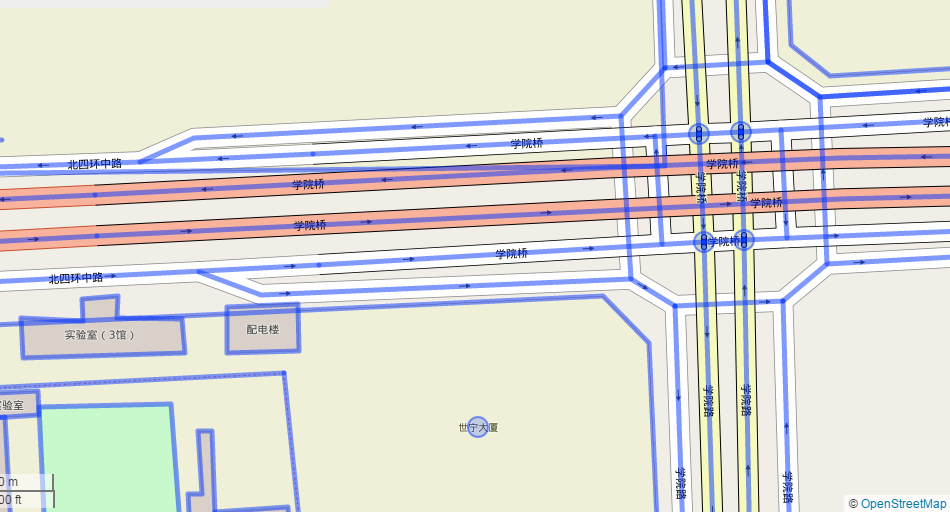


图3 北四环学院桥附近真实道路网络

图3是北京市北四环学院桥附件真实道路网络情况，从图中可以清晰地看到，四环和学院桥都有多条道路（主路、辅路、不同方向的道路段）组成。

OpenStreetMap情况

OpenStreetMap官方指导文档： <http://download.csdn.net/detail/scy411082514/4251597>

道路匹配情况

以目前系统车辆运行实际情况来看，车辆在北京市道路网路运行比较复杂，随机从系统抽取出了京QX0E92（devicesn为967790210497）的运行轨迹（在2015-12-28 20:40:06到2015-12-28 21:08:39时间段内）。



图4 某辆车某一运行轨迹图

该车在四环方向明显存在着两条路径，而且在不同路段匹配结果不一样。左边匹配不清楚，右边匹配结果较好。

表一 某辆车在不同路段匹配情况表

|  |
| --- |
| 某辆车不同路段匹配情况表 |
|  |
|  |
|  |

北四环有多条道路，用原始位置的匹配很难知道该车到底在哪条路上行驶、运动方向等情况。

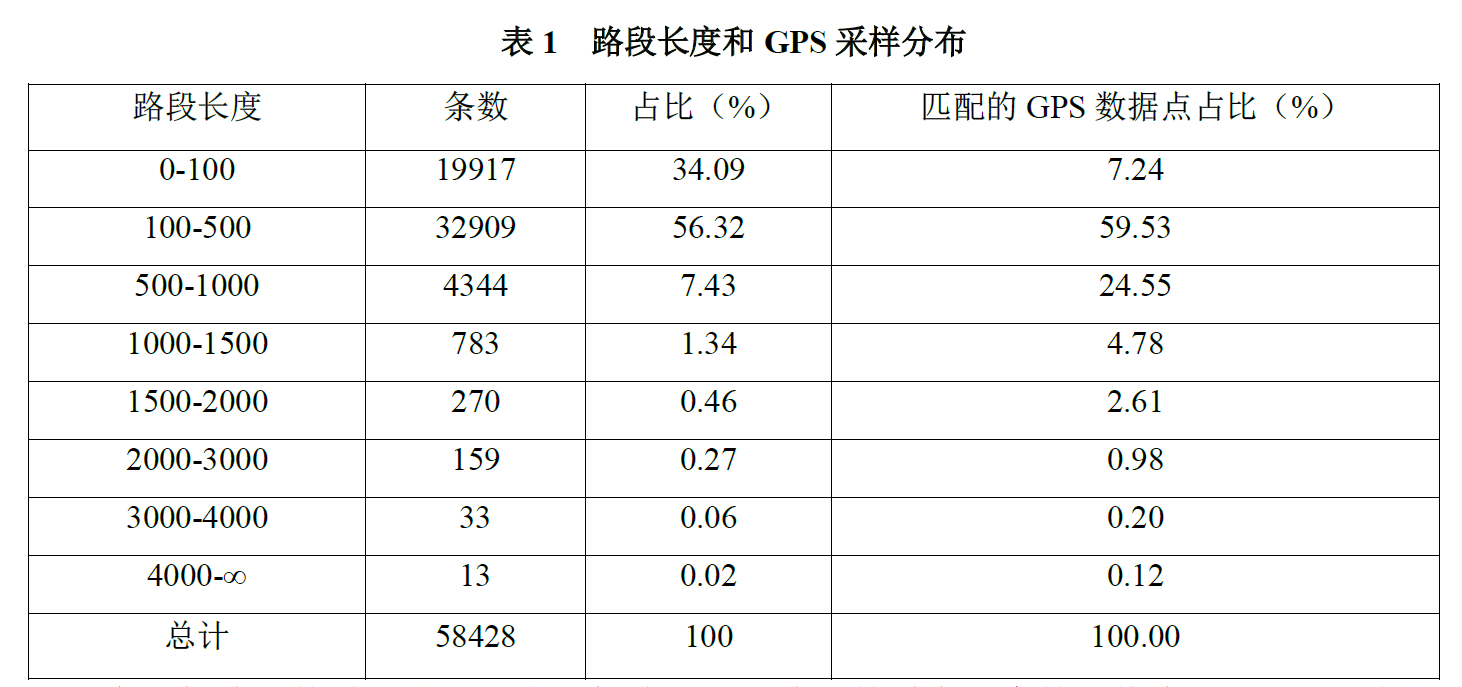
表二 某辆车在环形路口匹配情况表

|  |
| --- |
| 该车原始轨迹图 |
|  |
| 该车环形路口实际轨迹点图 |
|  |
| 该环形路口实际数据点图 |
|  |
|  |

该车在四环出口路段匹配不准确，在环形路段上匹配结果还可以（环形路段上有七个点，预计结果还可以）。

[Map-Matching for Low-Sampling-Rate GPS Trajectories]使用的地图为58624个点和130714个道路段（一条道路可以被分为很多个路段）。

杨晓宇使用了OpenStreetMap的shapefile文件，并统计了路段的长度等情况。



获取osm数据的方法：①官网页面直接选定，下载（小范围数据）。②wget，curl方法。具体链接：其中，坐标点为左下点（经度，纬度），右上点（经度，纬度）。

小范围：

<http://api.openstreetmap.org/api/0.6/map?bbox=11.54,48.14,11.543,48.145>

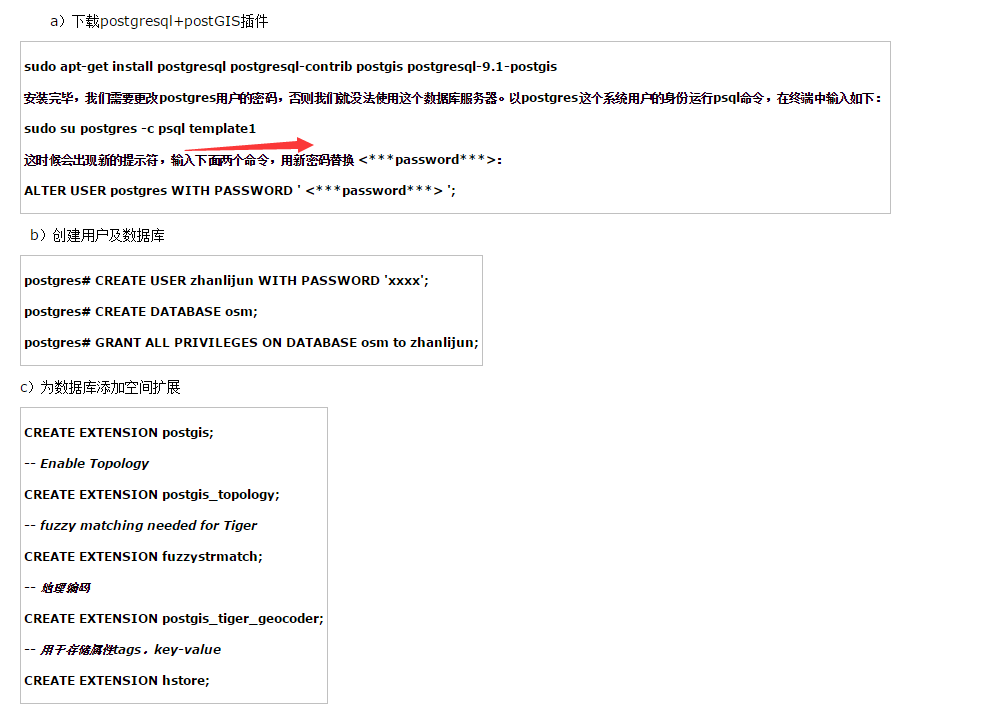
大范围：

http://overpass.osm.rambler.ru/cgi/xapi\_meta?\*[bbox=11.5,48.1,11.6,48.2]

大范围数据。

具体可参考说明链接：<http://wiki.openstreetmap.org/wiki/Downloading_data>

OSM环境、数据构建

1. 下载数据：在OSM官网下载格式xml或者sharpfile的文件，具体区域为（116.199814——116.554159，39.750665——40.027734）。
2. 安装postgresql数据库和postgis扩展。本文使用postgresql9.4、postgis2.0软件，操作系统环境为debian Wheezy7.5。具体为执行apt.postgresql.org.sh更改postgresql安装源（由于默认apt-get使用的是9.1版本），之后执行apt-get install postgresql-9.4。安装完成之后执行apt-get install postgis。安装后，记得修改/etc/postgresql/9.4/main目录中的pg\_hba.conf、postgresql.conf文件中的监听等选项，并重启。默认数据库的文件路径为/var/lib/postgresql/9.4/main/base。之后，更改密码、创建数据库等（记得一定在新创建的数据库中添加gis扩展，命令为create extension postgis）。
3. 下载osm2pgsql 工具，并将数据导入（如果使用postgres用户进入psql命令时，记得给导入的xml文件加入权限），执行类似命令：osm2pgsql -s -U postgres -d osm /tmp/map.xml -H 192.168.6.133 -W;。
4. 数据导入pg数据库成功，执行gis操作。  
   

更改安装源文件请见附件或者从官网下载。

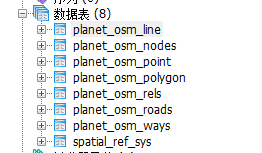
构建地图部分细节可以参考博客：<http://www.cnblogs.com/LBSer/p/4451471.html>

构建自己的地图服务器系列博客：http://blog.csdn.net/goldenhawking/article/details/7952303

Pgsql用户问题：<http://www.ruanyifeng.com/blog/2013/12/getting_started_with_postgresql.html>

导入后数据库和数据表列表：

目前数据库表格主要为：



其中，planet\_osm\_nodes、planet\_osm\_ways、planet\_osm\_rels为临时数据表，不用关注。

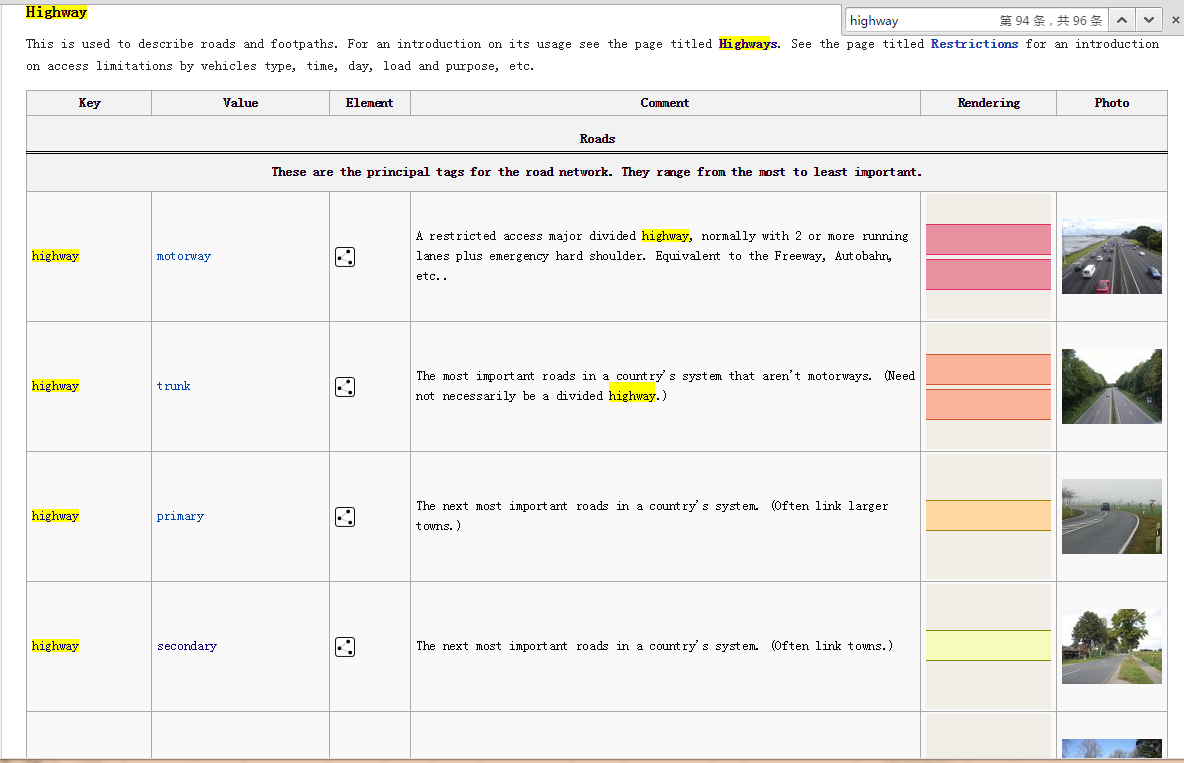
主要有四个数据表需要关注（主要是roads是line的子集，有些道路只在line里面）

* planet\_osm\_line: contains all imported ways
* planet\_osm\_point: contains all imported nodes with tags
* planet\_osm\_polygon: contains all imported polygons. Relations seem to be resolved for that.
* planet\_osm\_roads: contains a subset of planet\_osm\_line suitable for rendering at low zoom levels. planet\_osm\_line contains too many elements to render on overview maps.

每个表的具体解释可见以下网址：

<http://wiki.openstreetmap.org/wiki/Osm2pgsql/schema>

道路有不同的等级和分类，具体可以见以下地址：



<http://wiki.openstreetmap.org/wiki/Map_Features>

道路的具体属性可以见以下地址：

<http://wiki.openstreetmap.org/wiki/Highways>

数据库说明：

服务器地址为：192.168.6.133，用户名：postgres，密码：postgres

其中，osm为地图数据库，其经过postgis扩展。

gps\_data为抽取的北京市五环（含）的3月份到6月份的全部车辆数据，其中，每个表对应的是devicesn结尾数字的记录。

训练数据建立

从HBase中获取从3月份到6月份的北京五环内的原始数据作为历史数据，挖掘频繁模式。

准备做法：（初步考虑使用mapreduce运行）

1. 以每辆车作为一个job，选出3月到6月的数据，按照原始数据的gpstime排序，过滤范围后，存入关系型数据中（主要保存devicesn、gpstime、latitude、longitude、speed等）。

主要考虑问题：

1. 轨迹切分，获取原始数据后，需要得到每个运行轨迹，可参考系统的做法（即速度为0，时长在1分钟即可认为为一个轨迹段）。
2. 在获取原始轨迹点，注意按照实际GPS时间戳排序，防止出现乱序情况。考虑一个车一天至多一万个点，可以对一辆车所有的记录排序后再写入pgsql。



**数据处理过程中用到的SQL语句(Postgres下)**

**多表查询加更新：**

update new\_roads nr

set nodes = s.n

from (select id, planet\_osm\_ways.nodes as n from planet\_osm\_ways,new\_roads

where new\_roads.osm\_id=planet\_osm\_ways.id) as s

where nr.osm\_id = s.id

**将更低级的路加入new\_roads表中（先insert再update）：**

insert into new\_roads(osm\_id,nodes)

select id,nodes from planet\_osm\_ways where 'tertiary'=any(tags)

update new\_roads nr

set class=5,highway='tertiary'

from (select id,nodes as n from planet\_osm\_ways where 'tertiary'=any(tags)) as s

where nr.osm\_id=s.id

（注：set后面的字段不能加nr.）

**查重：**

select \* from new\_roads

where osm\_id in (select osm\_id from new\_roads group by osm\_id having count(osm\_id) > 1)

select distinct osm\_id, count(\*) from new\_roads group by osm\_id having count(\*) > 1;

**删除重复记录：**delete from new\_roads where ctid not in (select min(ctid) from new\_roads group by osm\_id);（PostgreSQL用ctid，其他有些数据库用rowid）

**将数据类型为数组的nodes字段拆分成单个元素存入新表：**

Insert into new\_nodes(id)

select unnest(nodes) from new\_roads

**利用way字段得到点的经纬度:**

select ST\_AsText(ST\_Transform(way,4326)) from planet\_osm\_point limit 10；

**将osm数据库中的lat和lon字段转化成经纬度:**

select ST\_AsText(st\_transform(st\_geomfromtext('POINT (12956614.45 4852286.48)',900913),4326))

**将new\_nodes表里的经纬度坐标转化成WSG84格式**

update new\_nodes n set longitude = s.longi, latitude = s.lati from

(select id,ST\_X(ST\_AsText(st\_transform(st\_geomfromtext('POINT ('||lon/100||' '||lat/100||')',900913),4326))) as longi,

ST\_Y(ST\_AsText(st\_transform(st\_geomfromtext('POINT ('||lon/100||' '||lat/100||')',900913),4326))) as lati from new\_nodes) as s

where n.id=s.id

**起始点id存入数组格式的数据中：**

update new\_road\_segments set nodes=Array[start\_id,end\_id]

**数据再处理用到的SQL语句：**

Insert into new\_roads\_sixclass(road\_id,highway,way,road\_class,nodes)

select road\_id,highway,way,class,nodes from new\_roads where class<=4

select new\_roads\_sixclass.road\_id,new\_roads\_sixclass.highway,new\_roads\_sixclass.way,new\_roads\_sixclass.class,new\_roads\_sixclass.nodes from new\_roads,new\_roads\_sixclass where new\_roads\_sixclass.road\_id =new\_roads.road\_id and new\_roads\_sixclass.class>=7

Insert into new\_nodes\_sixclass(node\_id)

select unnest(nodes) from new\_roads\_sixclass where road\_class<=4

update new\_nodes\_sixclass nr

set lat=s.lat,lon=s.lon,latitude=s.latitude,longitude=s.longitude,tags=s.tags,way=s.way,node\_class=s.class

from (select \* from new\_nodes) as s

where nr.node\_id=s.node\_id

select \* from new\_nodes\_sixclass order by node\_id

delete from new\_nodes where node\_id not in (SELECT distinct unnest(nodes) FROM new\_roads)

select distinct class from new\_roads group by class

select \* from planet\_osm\_roads where highway is not null and way is not null

update new\_roads nr

set way=s.way

from (select osm\_id,way from planet\_osm\_roads) as s

where nr.road\_id=s.osm\_id and nr.way is null

常用的坐标系：

WGS-84：是国际标准，GPS坐标（Google Earth、OpenStreetMap使用、或者GPS模块）

GCJ-02：中国坐标偏移标准，GoogleMap、高德、腾讯使用

BD-09：百度坐标偏移标准，BaiduMap使用

创建和另一个表一样字段的表

create table road\_ROADER (like road)

将一个表复制到另一个表

insert into road\_ROADER select \* from road

给数据表设置主键

alter table road\_roader add primary key (road\_id)