



# 智慧物流PostGIS应用

何祖文

贵州云飞科技有限公司

# 分享内容



01

要点

应用业务  
场景

02

要点

GPS介绍

03

要点

GIS应用

04

要点

百度地图  
结合

05

要点

GIST索引  
查询优化



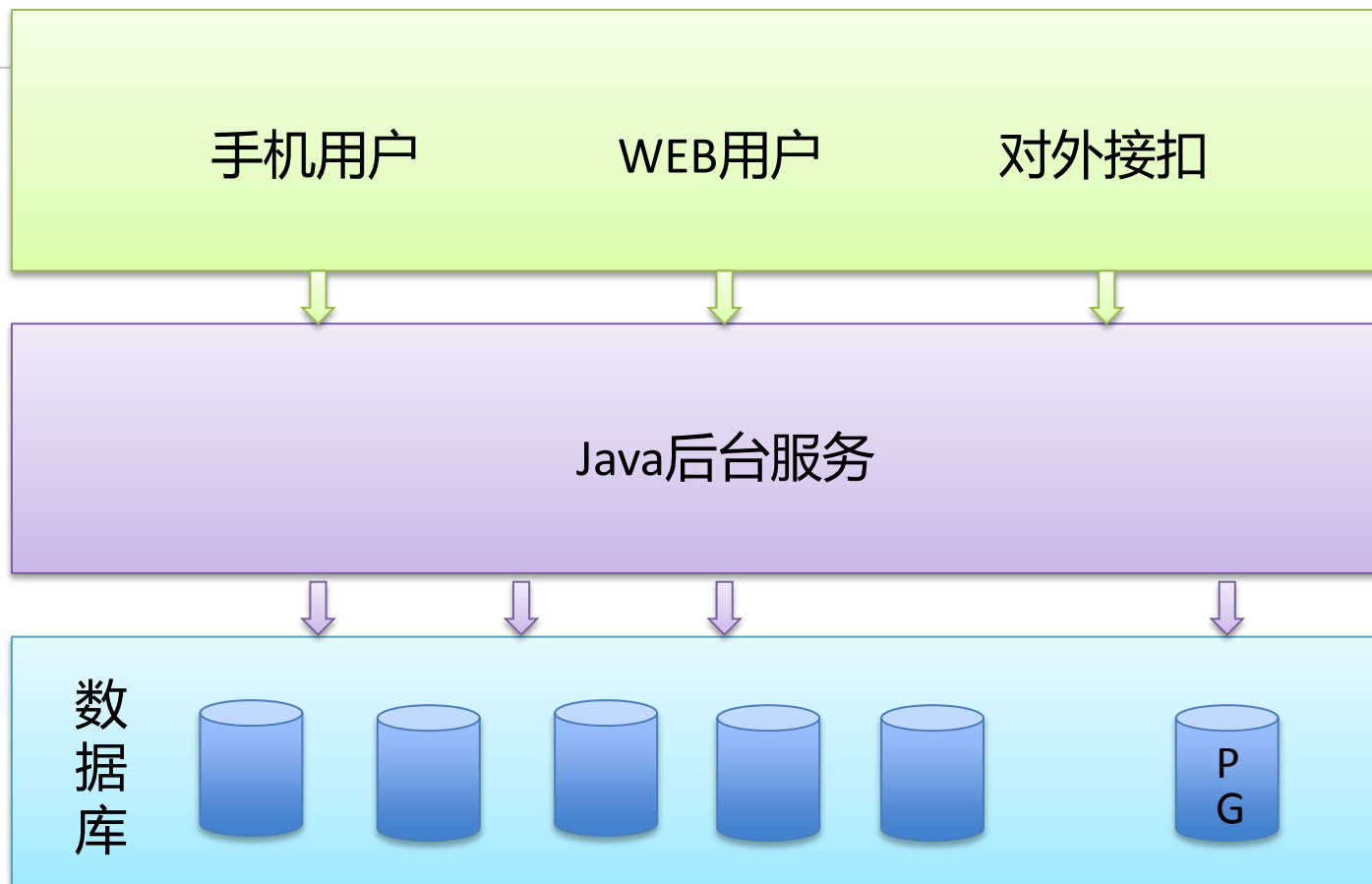
## 应用背景

目前国内的中小型物流企业运输现状，业务杂乱，货物难以追踪甚至有追不到的情况。针对这种特殊的情况，我们做出了物流运输平台。其中平台的运输模块数据主要使用PG来进行存储，货物车辆的追踪，历史线路的轨迹回放查询。





# 架构概述



# 最终实现



显示异常情况

在客户查询时显示



沿途采集车辆的GPS点



点击收发货点可查看收发货的详细信息

# GPS



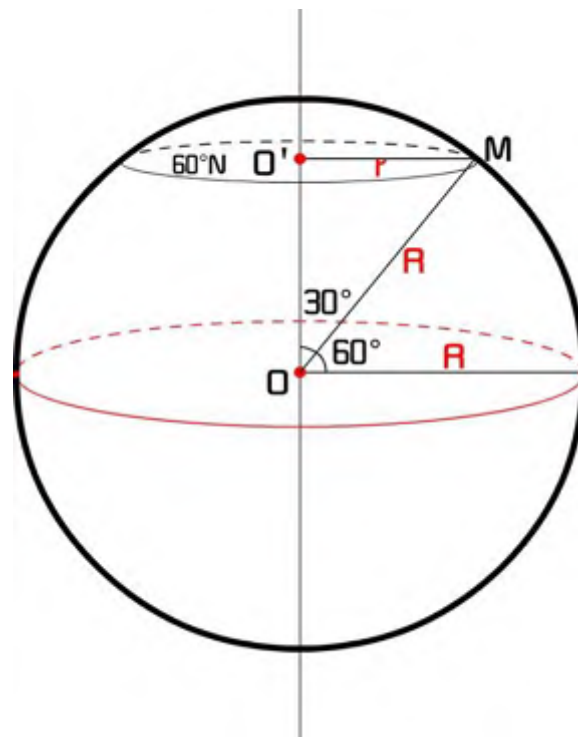
- GPS是英文Global Positioning System（全球定位系统）的简称。GPS起始于1958年美国军方的一个项目，1964年投入使用。
- 坐标系：常用的是 LAT/LON，LAT/LON 就是经纬度表示。
- 地图基准：一般用 WGS84。



# 赤道



地球是在不停地绕地轴旋转（地轴是一根通过地球南北两极和地球中心的假想线）在地球中腰画一个与地轴垂直的大圆圈，使圈上的每一点都和南北两极的距离相等 这个圆圈就叫“赤道”



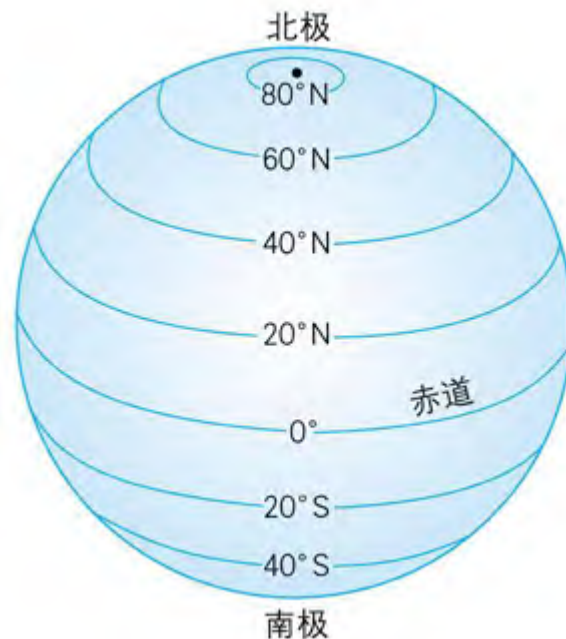
# 纬圈



定义为地球表面某点随地球自转所形成的轨迹。

在赤道的南北两边，画出许多和赤道平行的圆圈，就是“纬圈”。

我们把赤道定为纬度零度，向南向北各为90度，在赤道以南的叫南纬，在赤道以北的叫北纬。



北极就是北纬90度，南极就是南纬90度。纬度的高低也标志着气候的冷热，如赤道和低纬度地地区无冬，两极和高纬度地区无夏，中纬度地区四季分明。



# 经圈



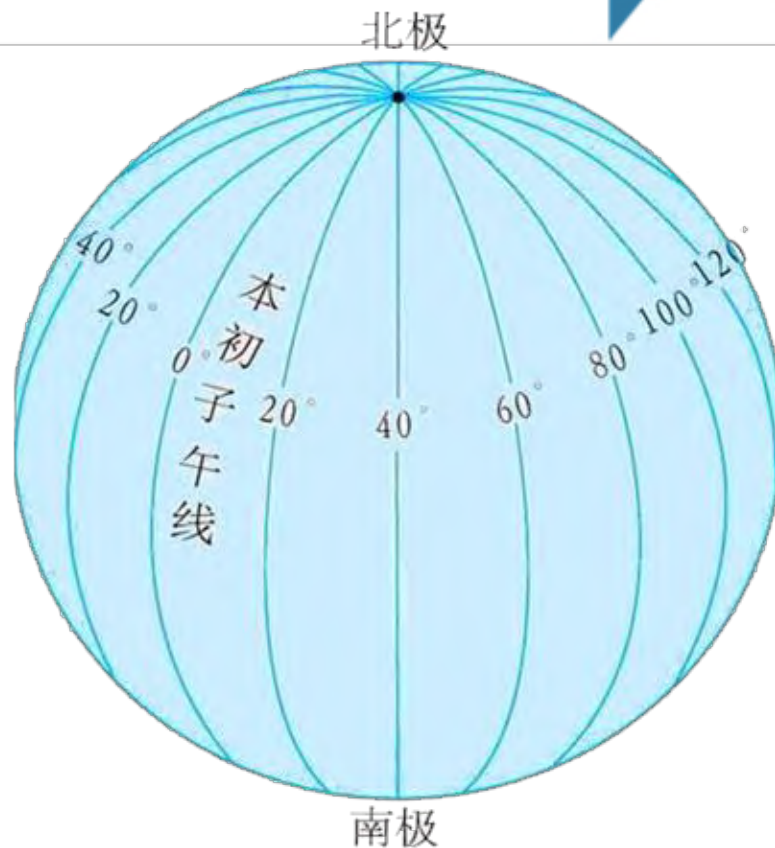
从北极点到南极点，可以画出许多南北方向的与地球赤道垂直的大圆圈，这叫作“经圈”；构成这些圆圈的线段，就叫经线。



公元1884年国际规定以通过英国伦敦近郊的格林尼治天文台的经线作为计算经度的起点，即经度零度零分零秒，也称“本初子午线”。在它东面的为东经，共180度

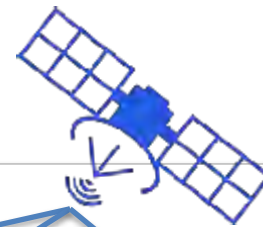


在它西面的为西经，共180度。因为地球是圆的，所以东经180度和西经180度的经线是同一条经线。各国公定180度经线为“国际日期变更线”。





# GPS存储



## 主要存储数据

1. GPS信息(point)
2. 车辆信息
3. 线路信息(line)
4. 行政区域信息



## 存储存在的问题——技术

1. GPS信号采集频率
2. GPS信号间断
3. 网络问题



## 存储存在的问题——业务

1. 司机不配合
2. 条件不具备



## 车辆的行驶路程计算公式

行驶的路程

根据行驶的GPS点来计算，下面是两点直接的距离换算公式。

计算公式

理论上的估算值，设第一点A的经纬度为(LonA, LatA)，第二点B的经纬度为(LonB, LatB)，按照0度经线的基准，东经取经度的正值(Longitude)，西经取经度负值(-Longitude)，北纬取90-纬度值(90- Latitude)，南纬取90+纬度值(90+Latitude)，则经过上述处理过后的两点被计为(MLonA, MLatA)和(MLonB, MLatB)。那么根据三角推导，计算两点距离的如下公式：

$$C = \sin(\text{LatA}) * \sin(\text{LatB}) + \cos(\text{LatA}) * \cos(\text{LatB}) * \cos(\text{MLonA} - \text{MLonB})$$
$$\text{Distance} = R * \text{Arccos}(C) * \text{Pi} / 180$$

# Google地图 提供的方法



1. Lat1 Lung1 表示A点经纬度，Lat2 Lung2 表示B点经纬度；  
2. a=Lat1 - Lat2 为两点纬度之差 b=Lung1 - Lung2 为两点经度之差

```
private static final double EARTH_RADIUS = 6378137; //赤道半径(单位m)
/** * 转化为弧度(rad) * */
```

```
private static double rad(double d) { return d * Math.PI / 180.0; }
```

```
/**
```

```
* 基于googleMap中的算法得到两经纬度之间的距离
```

```
* @param lon1 第一点的经度
```

```
* @param lat1 第一点的纬度
```

```
* @param lon2 第二点的经度
```

```
* @param lat2 第二点的纬度
```

```
* @return 返回的距离，单位km
```

```
*/
```

```
public static double GetDistance(double lon1, double lat1, double lon2, double lat2)
```

```
{
```

```
    double radLat1 = rad(lat1);
```

```
    double radLat2 = rad(lat2);
```

```
    double a = radLat1 - radLat2;
```

```
    double b = rad(lon1) - rad(lon2);
```

```
    double s = 2 * Math.asin(Math.sqrt(Math.pow(Math.sin(a/2), 2) + Math.cos(radLat1) * Math.cos(radLat2) * Math.pow(Math.sin(b/2), 2)));
```

```
    s = s * EARTH_RADIUS;
```

```
    //s = Math.round(s * 10000) / 10000;
```

```
    return s; }
```

$$S = 2 \arcsin \sqrt{\sin^2 \frac{a}{2} + \cos(\text{Lat}1) \times \cos(\text{Lat}2) \times \sin^2 \frac{b}{2}} \times 6378.137$$



## 使用double字段 存储GPS

### 存储数据

```
create table sys_gps(  
  `id` bigint `id主键`,  
  `position_name` varchar(225) COMMENT '地点名称',  
  `lng` double(20,6) COMMENT '经度',  
  `lat` double(20,6) COMMENT '纬度',  
  `car_id` bigint COMMENT '车辆编号',  
  `get_time` varchar(30) COMMENT '获取时间',  
  PRIMARY KEY (`id`));
```



### 插入数据

```
insert into sys_gps values('1','贵州省贵阳市贵阳高新技术开发区云飞科技',  
'106.661412,26.626848,10883','2016-10-10 14:42:45');  
insert into sys_gps values('2','贵州省贵阳市贵阳高新技术开发区白云大道',  
'106.662885,26.62638,10883','2016-10-10 14:43:45');  
insert into sys_gps values('3','贵州省贵阳市贵阳高新技术开发区阳光大道',  
'106.661412,26.628835,10883','2016-10-10 14:44:45');
```



# SQL查询



SQL计算云飞科技到白云大道的距离(单位米)

```
select 6378.137 * 2 * asin(sqrt(pow(sin( (26.626848 * pi() / 180 - 26.62638 * pi() / 180) / 2, 2) + cos(26.626848 * pi() / 180) * cos(26.62638 * pi() / 180) * pow(sin( (106.661412 * pi() / 180 - 106.662885 * pi() / 180) / 2, 2))) * 1000 distance;
```

```
+-----+
```

```
| distance |
```

```
+-----+
```

```
| 155.56637542490483 |
```

```
+-----+
```

百度地图API计算距离(单位米) 155.39



## 自定义查询 两点距离方法



```
/**
 * lng1 lat1表示A点经纬度
 * lng2 lat2表示B点经纬度
 **/
CREATE DEFINER='root'@'%' FUNCTION `get_distance`(lng1 double,lng2
double,lat1 double ,lat2 double) RETURNS double
BEGIN
    DECLARE d double;
    select 6378.137 * 2 * asin(sqrt(pow(sin( (lat1 * pi() / 180 - lat2 * pi() /
180) / 2), 2) + cos(lat1 * pi() / 180) * cos(lat2 * pi() / 180) * pow(
sin( (lng1 * pi() / 180 - lng2 * pi() / 180) / 2), 2))) * 1000 into d;
    return d;
END
```

## SQL查询距离 某点N米的车辆



```
select get_distance(t.lng,106.662885,t.lat,26.62638) distance,t.car_id  
from sys_gps t where get_distance(t.lng,106.662885,t.lat,26.62638) < 1000  
and t.id in (select max(s.id) from sys_gps s group by s.car_id);
```

```
+-----+-----+  
| distance      | car_id |  
+-----+-----+  
| 310.11842991849755 | 10883 |  
| 155.37885479970285 | 10884 |  
| 171.38145468051675 | 10776 |  
| 499.08698305396814 | 13003 |  
| 554.7571559396379  | 11032 |  
| 572.9669592026722  | 16954 |  
| 113.25176126509592 | 11365 |  
+-----+-----+
```



# GIS 简介



## - 定义

地理信息系统 ( Geographic Information System或 Geo - Information system , GIS ) 有时又称为 “地学信息系统” ，是一个空间信息系统。

## - 运用

数据进行采集、储存、管理、运算、分析、显示和描述的技术系统。

## - 组成

人员，数据，硬件，软件，过程





## POSTGIS 简介

### - 定义

PostGIS是对象关系型数据库系统PostgreSQL的一个扩展，PostGIS提供如下空间信息服务功能：空间对象、空间索引、空间操作函数和空间操作符。同时，PostGIS遵循OpenGIS的规范。

### - PostGIS

2001年的5月发布了PostGIS的第一版 ( PostGIS V0.1 )

### - 特性

空间数据类型 ( 点、线、多边形、多点、多线... )

数据存取和构造方法(GeomFromText()、AsBinary()...)

空间分析函数(Area和Length...)

元数据的支持

二元谓词

空间操作符

...





## POSTGIS 类型

POINT(0 0) ——点

LINSTRING(0 0,1 1,1 2) ——线

POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1)) ——面

MULTIPOINT(0 0,1 2) ——多点

MULTILINSTRING((0 0,1 1,1 2),(2 3,3 2,5 4)) ——多线

MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1))) ——多面

GEOMETRYCOLLECTION(POINT(2 3),LINSTRING((2 3,3 4))) ——

几何集合

## 几何对象关系函数



获取两个几何对象间的距离 `ST_Distance(geometry, geometry)`

判断两个几何对象是否分离 `ST_Disjoint(geometry, geometry)`

判断两个几何对象是否相交 `ST_Intersects(geometry, geometry)`

判断两个几何对象的边缘是否接触 `ST_Touches(geometry, geometry)`

判断两个几何对象是否互相穿过 `ST_Crosses(geometry, geometry)`

判断A是否被B包含 `ST_Within(geometry A, geometry B)`

判断两个几何对象是否是重叠 `ST_Overlaps(geometry, geometry)`

判断A是否包含B `ST_Contains(geometry A, geometry B)`

判断A是否覆盖 B `ST_Covers(geometry A, geometry B)`

判断A是否被B所覆盖 `ST_CoveredBy(geometry A, geometry B)`

## 几何对象处理函数



- 获取几何对象的中心 ST\_Centroid(geometry)
- 面积量测 ST\_Area(geometry)
- 长度量测 ST\_Length(geometry)
- 返回曲面上的一个点 ST\_PointOnSurface(geometry)
- 获取边界 ST\_Boundary(geometry)
- 获取缓冲后的几何对象 ST\_Buffer(geometry, double, [integer])
- 获取多几何对象的外接对象 ST\_ConvexHull(geometry)
- 获取两个几何对象相交的部分 ST\_Intersection(geometry, geometry)
- 将经度小于0的值加360使所有经度值在0-360间  
ST\_Shift\_Longitude(geometry)
- 获取两个几何对象不相交的部分 ( A、 B可互换 )  
ST\_SymDifference(geometry A, geometry B)
- 从A去除和B相交的部分后返回 ST\_Difference(geometry A, geometry B)
- 返回两个几何对象的合并结果 ST\_Union(geometry, geometry)
- 返回一系列几何对象的合并结果 ST\_Union(geometry set)

## 几何对象存取函数



- 判断几何对象是否为空 ST\_IsEmpty(geometry)
- 判断几何对象是否不包含特殊点（比如自相交） ST\_IsSimple(geometry)
- 判断几何对象是否闭合 ST\_IsClosed(geometry)
- 判断曲线是否闭合并且不包含特殊点 ST\_IsRing(geometry)
- 获取多几何对象中的对象个数 ST\_NumGeometries(geometry)
- 获取多几何对象中第N个对象 ST\_GeometryN(geometry,int)
- 获取几何对象中的点个数 ST\_NumPoints(geometry)
- 获取几何对象的第N个点 ST\_PointN(geometry,integer)
- 获取线的终点 ST\_EndPoint(geometry)
- 获取线的起始点 ST\_StartPoint(geometry)
- 获取几何对象的类型 GeometryType(geometry)
- 获取点的X坐标 ST\_X(geometry)
- 获取点的Y坐标 ST\_Y(geometry)
- 获取点的Z坐标 ST\_Z(geometry)
- 获取点的M值 ST\_M(geometry)

# GIS



## - 接触

最开始接触POSTGIS，是因为我们要对空间信息进行处理，过程中慢慢的团队开始接触POSTGIS

## - 查询之痛

随着我们业务数据量的增长，我们的查询效率越来越慢

## - 部分转移

决定将原系统中的空间信息部分转移到POSTGIS来





## 存储 Point

```
create table sys_gps(id bigint,car_id bigint, position_name
varchar(20),point_xy geometry);

INSERT INTO sys_gps (id,car_id, position_name,point_xy) VALUES
(1,100001,'北京',ST_Transform(ST_GeomFromText('POINT(116.46 39.92)',
4326), 2163));
INSERT INTO sys_gps (id,car_id, position_name,point_xy) VALUES
(2,100002,'上海', ST_Transform(ST_GeomFromText('POINT(121.47 31.23)',
4326), 2163));
INSERT INTO sys_gps (id,car_id, position_name,point_xy) VALUES
(3,520100,'贵阳', ST_Transform(ST_GeomFromText('POINT(106.71 26.57)',
4326), 2163));
INSERT INTO sys_gps (id,car_id, position_name,point_xy) VALUES
(3,520100,'乌鲁木齐', ST_Transform(ST_GeomFromText('POINT(87.6177
43.7928)', 4326), 2163));
```

.....





## GIS函数-距离

点跟点的直线距离——贵阳到乌鲁木齐(单位千米)

```
SELECT ST_Distance(  
ST_Transform(ST_GeomFromText('POINT(106.71 26.57)', 4326), 2163),  
ST_Transform(ST_GeomFromText('POINT(87.6177 43.7928)', 4326), 2163))  
/ 1000 st_distance;
```

```
+-----+
```

```
| st_distance |
```

```
+-----+
```

```
| 3311.3406955291 |
```

```
+-----+
```

```
1 row in set
```



## GIS函数-范围

查询云飞科技周边10千米的车辆

```
select *,ST_Distance(point_xy, ST_Transform(
    ST_GeomFromText('POINT(106.661412 26.626848)', 4326), 2163))
from sys_gps where point_xy &&
ST_Buffer(ST_Transform(ST_GeomFromText('POINT(106.661412 26.626848)',
    4326), 2163), 10000, 10);
```

```
+-----+
| car_id | st_distance   |
+-----+
| 10036  | 1016.9653164 |
| 28643  | 3649.89641365|
| 29751  | 8964.623     |
+-----+
```

3 row in set



驱动下载 : <http://jdbc.postgresql.org/download.html>

JDBC

```
Class.forName("org.postgresql.Driver");  
Connection connection = DriverManager.getConnection(  
"jdbc:postgresql://***.***.***.***:5432/**", "***", "***");
```

Hibernate 配置

```
<property name="connection.driver_class">org.postgresql.Driver</property>  
<property  
name="connection.url">jdbc:postgresql://***.***.***.***/**</property>;  
<property name="connection.username">***</property>  
<property name="connection.password">***</property>
```

PG的Java对象

PGpoint  
Pgline .....

## PG-GPS采集



APP端

负责采集GPS点信息并上传到Java端。

服务端

接收APP端上传的GPS采集信息数据，并入到PG数据库。

PG端

存储GPS信息。



## 轨迹回放



PG端

做GPS轨迹信息查询的优化。

Java端

结合PG提供的PgPoint对象读取PG存储的GPS点信息。

Web端

结合百度地图的API(Polyline对象)实现将后台查询出来的GPS轨迹信息绘制在地图上。



## GiST索引 查询优化



### GiST

GiST的全称是“通用搜索树”，是索引的一般形式。

优  
化

### 创建GiST索引

```
create index  
idx_cust_car_gps_1  
on area_gps using  
gist(point_xy);
```

### 建立GiST索引的语法

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

上面的语法是将建立2D索引。要建立PostGIS2.0+支持的n维索引，你可以用下面的语法：

```
CREATE INDEX [indexname] ON [tablename] USING GIST ([geometryfield]  
gist_geometry_ops_nd);
```

## GIS的后续



Point 3D点

存储室内数据

Line

转移原系统的线路数据

Web Site

部分业务直接集成到 PostgreSQL 来，应用程序直接来访问 PostgreSQL Web 接口获取数据，如车辆的GPS点数据



# Thanks!

## Q & A