

# **Building Location Aware Apps - Get Started with PostGIS**

## **PART II**

**Lasma Sietinsone**  
**[l.sietinsone@ed.ac.uk](mailto:l.sietinsone@ed.ac.uk)**



# Code

<http://gismatic.com/dev8d/scripts/>

# pgAdmin III – GUI interface for PostgreSQL

The screenshot displays the pgAdmin III interface. On the left, the Object browser shows a tree structure: Server Groups > Servers (2) > pg\_intro (localhost:5432) > Databases (1) > pg\_intro > Schemas (1) > public > Tables (17). The 'amenity' table is selected. The main pane shows the Properties tab for this table.

Property	Value
Name	amenity
OID	19802
Owner	test
Tablespace	pg_default
ACL	
Of type	
Primary key	osm_id
Rows (estimated)	0
Fill factor	
Rows (counted)	0

Below the Properties tab is the SQL pane, which contains the following SQL code:

```
-- Table: amenity
-- DROP TABLE amenity;

CREATE TABLE amenity
(
    osm_id integer NOT NULL,
    amenity text,
    name text,
    operator text,
    tags hstore,
    geog geography,
    geom geometry,
    CONSTRAINT amenity_pkey PRIMARY KEY (osm_id ),
    CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2),
    CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 4326)
```

At the bottom of the window, a status bar indicates "Retrieving details on table amenity... Done." and a timing of "0.23 secs".

# psql - PostgreSQL interactive terminal

Powerful but need to know commands!

```
Command Prompt - psql -h localhost -p 5432 -d pg_intro -U test

C:\>psql -h localhost -p 5432 -d pg_intro -U test
psql (9.1.2)
WARNING: Console code page (775) differs from windows code page (1257)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

pg_intro=> \x
Expanded display is on.
pg_intro=> select * FROM amenity LIMIT 1;
-[ RECORD 1 ]-----
osm_id      | 462025325
amenity     | bank
name        | Halifax
operator    | Halifax
tags        | "atm"=>"yes", "name"=>"Halifax", "amenity"=>"bank", "operator"=
>"Halifax"
geog        | 0101000020E61000004E208E1E09A5C3BF7F5DC3DAB7C14940
geom        | 0101000020E61000004E208E1E09A5C3BF7F5DC3DAB7C14940

pg_intro=>
pg_intro=>
```

# PostGIS – quick show around

- Functions start with ST\_\*
- Metadata tables:
  - geometry\_columns
  - spatial\_ref\_sys
- Utilities:
  - Shp2pgsql (import)
  - Pgsq2shp (export)



# EXERCISE 1 - import ESRI Shapefile

- File: greater\_london\_const\_region.shp
- *shp2pgsql, shp2pgsql-gui*: load SHP, DBF files

```
$ shp2pgsql --help
$ shp2pgsql [options] [shapefile schema.table | psql
    -h [host] -p [port] -U [user] [database]
```
- Derive centroids:
  - Place labels on the centre point of my geometries
- Derive bounding boxes:
  - Get all Flickr photos that fall within each bounding box
- Dissolve/union shapes
- View data with QGIS



# Desktop GIS Tools – Fun!

- View database contents (QGIS, OpenJUMP)
- Create on-the-fly maps (OpenJump)

**QGIS** – Quantum GIS

<http://www.qgis.org/>



**OPENJump**

<http://www.openjump.org/>



EDINA®



## EXERCISE 2 - import CSV file with postcodes

- File: wc.csv
- Standard formats: CSV, TEXT

```
$ psql --help
```

```
$ psql -h [host] -p [port] -U [user] -c "COPY [tablename]  
FROM [filename] WITH CSV DELIMITER ',' [database]"
```

- Make geometry from latitude & longitude (northing & easting)
- Set spatial reference system
- Add spatial index
- Cluster data





## EXERCISE 3 - import OSM

- *osm2pgsql*: load OSM data

```
$ osm2pgsql --help
```

```
$ osm2pgsql [options] planet.osm
```

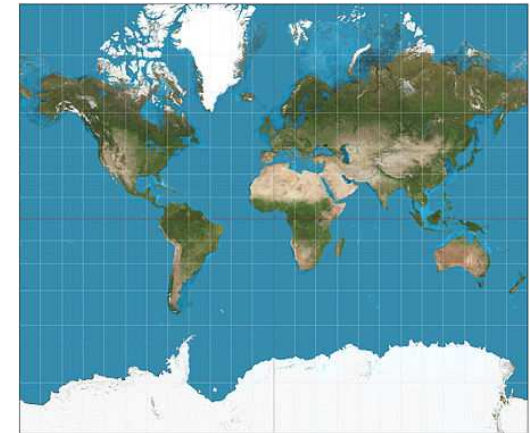
```
$ osm2pgsql [options] planet.osm.{gz,bz2}
```

```
$ osm2pgsql [options] file1.osm file2.osm file3.osm
```

- Query tags
- Geometry in text, geojson and other formats
- Align data with different SRS – projections & transformations!

# Map projections (SRS, SRID)

- **Conformal** – preserve angle
  - WGS84 (EPSG:4326) – Google Earth
  - OSGB 1936 (EPSG:27700) - British National Grid
  - *Google Projection ("EPSG:900913) – Google Maps*
- **Equal-area** – preserve area
- **Equidistance** – preserve distance for some standard line



<http://spatialreference.org>

## EXERCISE 4 – backend for “POIs” app



***OSM based nearest POI  
(Point of Interest)  
queries for location  
enabled mobile device***

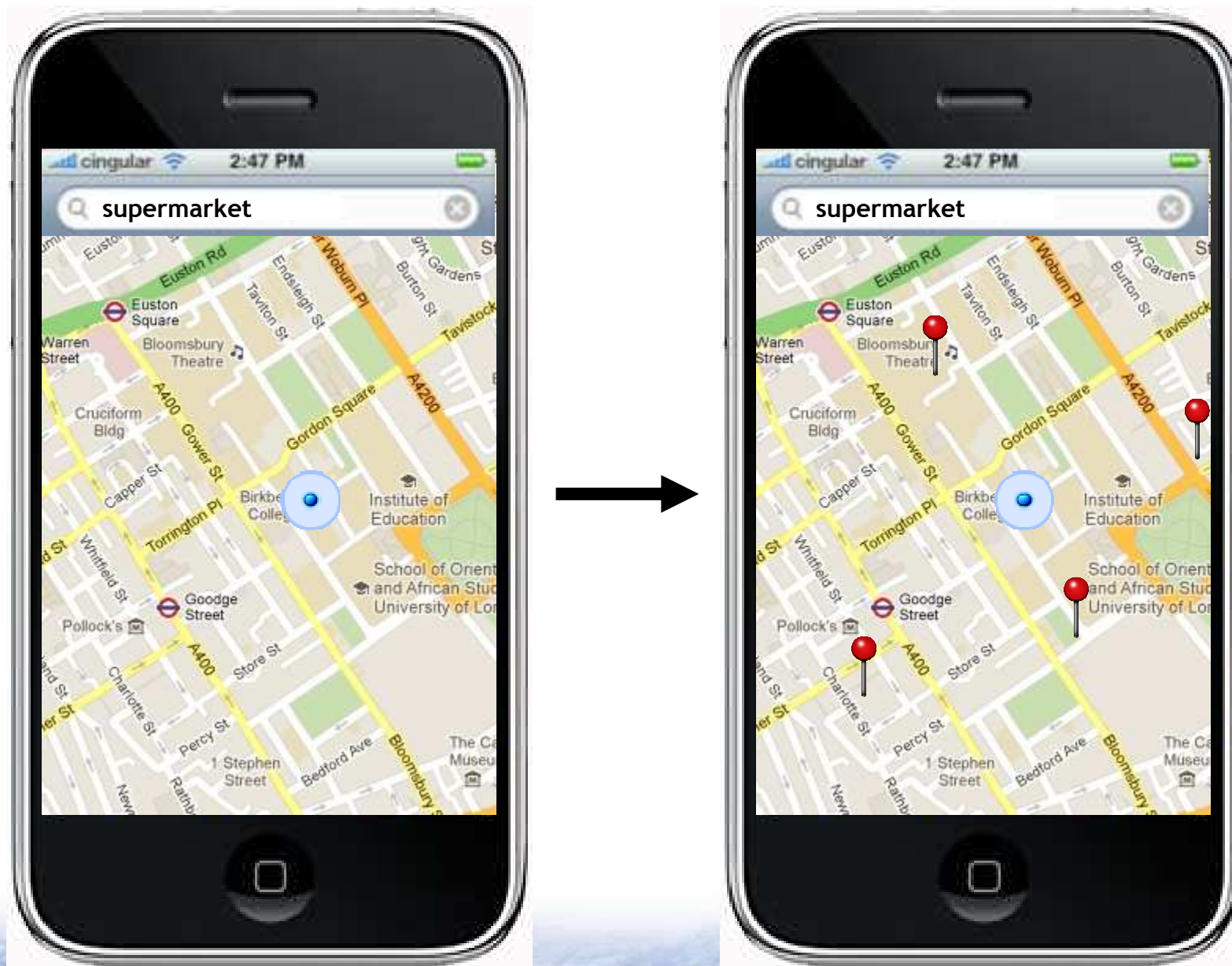
*e.g.*

*pub? (-> food?) -> cash? -> food?*

*supermarket??*

*hotel?*

## EXERCISE 4 – backend for “POIs” app (cont.)



## EXERCISE 4 – backend for “POIs” app (cont.)

- Create user tables for OSM POIs:
  - 08\_demo\_init\_poi\_tables.sql
- Create stored procedure for location queries:
  - 09\_demo\_stored\_procedure.sql



**Thank you!**

***Questions?***

