**Building a Geodatabase to understand the Siting of Countries’ Capitals**

Xiaohui Liu

A Tutorial on Design and Deployment of Databases and Geo-databases

Contents

[Abstract 3](#_Toc418545783)

[1. Introduction 4](#_Toc418545784)

[2. Methodology 4](#_Toc418545785)

[2.1 Dataset 4](#_Toc418545786)

[2.2 Software Selection 4](#_Toc418545787)

[2.3 Non-spatial Database Design 5](#_Toc418545788)

[2.3.1 E/R Model and Unified modeling language (UML) notation 5](#_Toc418545789)

[2.3.2 Relational and Database Schemas 7](#_Toc418545790)

[2.3.3 Functional dependencies 8](#_Toc418545791)

[3. Spatial Database Implementation 8](#_Toc418545792)

[4. Conclusions 13](#_Toc418545793)

[References 15](#_Toc418545794)

[Appendix 16](#_Toc418545795)

**Abstract**

The distribution pattern of capital cities worldwide varies greatly depending on various geographic, historical, or political reasons. This project aims to explore the distribution patterns of most capital cities worldwide relative to the shape of countries they pertain to. Specifically, the distance between countries’ capital and their geographic centroid are calculated to explore if there is any influence of countries’ shape on the siting of their capitals.

A geodatabase is an ideal way to organize both geographic and non-geographic dataset and provides the capability to perform spatial queries on objects that represent real-world geographic objects. This project demonstrates how to build both spatial and non-spatial databases for a real-life situation through all phases of database design, which consist of designing entity-relationship (E/R) model, UML notation, relational and database schemas, and functional dependencies for non-spatial database using Microsoft Access; constructing GIS file from textual data, building spatial queries, and measuring geographic distances for spatial database using PostgreSQL 9.4.

By implementing spatial database and execute spatial queries, sophisticated geographic relationships among the study objects can be identified to assist further understanding of geographic relationships.

1. **Introduction**

The distribution pattern of capital cities worldwide varies greatly depending on various geographic, historical, or political reasons. This tutorial presents an approach to constructing both spatial and non-spatial database and demonstrates spatial queries with the prototype geodatabase. The goal of this project is to build a geodatabase that documents the relationship between countries, capitals, and provinces, and specifically, try to explore if there is any influence of countries’ shape on the siting of their capitals.

1. **Methodology**
   1. Dataset

The data is acquired from a Microsoft Access sample database, which is originally extracted from the MONDIAL project (Chapple 2015). The sample database contains information on the world’s countries, cities, and provinces. For the sake of this project, the sample database was further extracted. The database contains three tables: country, city, and province. The country table consists of country name, country code, capital, province, area, and population fields; the city table contains city name, country, population, latitude, and longitude. The province table contains province name, country, population, and capital.

In order to implement the database spatially, the city table with latitude and longitude was used to create an Esri feature class. Together with a downloaded dataset of countries’ boundaries worldwide (ArcGIS Online 2014), the city feature class was used for spatial queries.

* 1. Software Selection

PostgreSQL 9.4, an advanced open source, object-oriented database management system is used to store the spatial data and perform spatial queries. PostgreSQL is chosen because it is free software and is publicly available, and has great extensibility and compliant standards with major proprietary and open source databases. Moreover, PostgreSQL implements the majority of the SQL 2011 standard and has a spatial database extension named PostGIS, which allows location queries to be executed in SQL (PostGIS 2015). Additionally, the compatibility with QGIS makes it possible to mapping, edit, and visualize location-based query results easily (QGIS 2015).

* 1. Non-spatial Database Design
     1. E/R Model and Unified modeling language (UML) notation

E/R model is a data model for describing data and the internal relationships between objects in the data. It contains entity sets, entity attributes, and relationship between entities (Kar, 2015). Entity and relationship are the main components of an E/R model. An entity represents an object that is distinguishable from other objects. An entity may or may not have attributes which indicates the property of the entity. Relationship shows the connections among two or more entities that constitute the E/R model. One or a certain combination of an entity’s attributes acts as the unique identifier, also called primary key, for each tuple (or row) in a relation. E/R model is usually demonstrated in the form of an E/R diagram, where rectangles stand for entity sets, ovals means attributes of entity sets, and diamonds denote the relationship between entity sets. The E/R diagram for this project is shown in Fig. 1 with primary key underlined.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\E_R Model.jpg |
| Fig. 1. E/R model |

UML is a type of object oriented model for representing data and their relationships (Kar 2015), which facilitates the visualization of the data in the database. Fig. 2 shows the UML for the database.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\UML.JPG |
| Fig. 2. UML diagram |

* + 1. Relational and Database Schemas

The tables modified based on the E/R model and UML diagram are described in the following section. The primary key of each table is underlined and each field is followed by data types of the attributes.

Country (Code: CHAR(10), CountryName: VARCHAR(15), Capital: VARCHAR(15), Area: DOUBLE, Population: LONG)

* Code: a short alphabetic code used as the abbreviation of the country’s name
* CountryName: the name of the country
* Capital: the name of the country’s capital
* Area: the area of the country
* Population: the population of the country

Province(Name: VARCHAR(15), Capital: VARCHAR(15), Area: DOUBLE, Population: LONG)

* Name: the name of the province
* Capital: the capital of the province
* Area: the area of the province
* Population: the population of the province

City(Name: VARCHAR(15), Population: LONG, Longitude: DOUBLE, Latitude: DOUBLE)

* Name: the name of the city
* Population: the population of the province
* Longitude: the longitude of the city
* Latitude: the latitude of the city

The schemas of the two relations are as follows:

Coun\_prov(CountryCode: CHAR(10), ProvinceName: VARCHAR(15))

Coun\_city (CountryCode: CHAR(10), City\_name: VARCHAR(15))

* + 1. Functional dependencies

Since code of country table, name of city table, and name of province table can exclusively get the rest of the attributes of each entity, code (country), name (city), and name (province) are chosen as the primary keys for each table and the rest of the attributes functionally depend on their primary key.

Both province and city table have a connection with the country table through a common field, Code, which is the primary key for country table. In this sense, code becomes the foreign key for both province and city table. Therefore, relation Coun\_prov, which denotes the relation between country and province, consists of the key of country and province table; relation Coun\_city, denotes the relation between country and city, consists of the key of country and city table.

1. **Spatial Database Implementation**

In order to perform spatial queries based on the database, city table with longitude/latitude was converted to Esri shapefile and then imported into PostgreSQL. Since country table does not contain geographic locational information, a shapefile of world countries’ boundary was downloaded from ArcGIS online (2014) and imported to PostgreSQL through PostGIS Shapefile and DBF Loader 2.1 in the Plug-ins drop-down box in the PG Admin III interface. Fig. 3 shows the process of importing country shapefile into PostgreSQL.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\2.JPG |
| Fig. 3 Importing shapefile into PostgreSQL |

Since no geographic locational informational is available in province table, it can only be imported as non-spatial information in the geodatabase. Fig. 4 shows the province table with its schema used to store the non-spatial information.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\13.JPG |
| Fig. 4 Province table |

PostgreSQL implements the majority of the SQL 2011 standard and have a user-friendly interface to write and execute queries. Fig. 5 shows the SQL statement of finding capital cities along with the query result.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\4.JPG |
| Fig. 5 Query for capital cities |

The goal of this project is to explore the distribution patterns of most capital cities worldwide relative to the shape of countries they pertain to. Specifically, the distance between countries’ capital and their geographic centroids needs to be calculated. Fig. 6 shows the SQL statement used to find the geometries of capital cities and countries’ geographic centroids. A view is created based on the previous query result and will serve as a base-relation for further queries. Fig. 7 shows the query for the distance between countries’ capital and their geographic centroids based on the prior saved view relation. Fig. 8 shows some statistics on the distances calculated in Fig. 7, with a maximum distance of 249.80 degree, a minimum distance of 0.02 degree, and an average distance of 9.92 degree.

|  |
| --- |
| C:\Xiaohui\GeoDB\project\9.JPG |
| Fig. 6 Query for geometries of capital cities and countries’ geographic centroids |

|  |
| --- |
| C:\Xiaohui\GeoDB\project\10.JPG |
| Fig. 7 Query for distance between countries’ capital and their geographic centroid |
|  |
| C:\Xiaohui\GeoDB\project\12.JPG |
| Fig. 8 Query for statistics on the distances |
|  |
| Fig. 9 Distance Between Countries' Capital and Centroid |

As is shown in Fig. 8, the average distance between countries’ capital and their geographic centroids is 9.92093 degree, which is equal to 1100.73394 km (arc length = radius \* central angle), which is about a third of the length of Japan. This number might not have universal significance and is not representative among all the countries because the shape and size of countries vary greatly. However, except a few countries with large distance between their capitals and centroids, most of the countries tend to have very short distances between the two. Fig. 9 presents the distances in Kilometers for the counties under study. The majority of the countries clustered in the short-distance region, which allows the distances to form a skewed distribution toward the minimum distance side. In this sense, we can see that capital cities tend to be located near to countries geographic centroid and not vice versa.

1. **Conclusions**

This tutorial presents an approach to exploring the distribution patterns of most capital cities worldwide relative to the shape of countries they pertain to. Specifically, by calculating the distance between countries’ capital and their geographic centroids to explore if there is any influence of countries’ shape on the siting of their capitals. This project demonstrates how to build both spatial and non-spatial databases for a real-life situation through all phases of database design, which consist of designing entity-relationship (E/R) model, UML notation, relational and database schemas, and functional dependencies for non-spatial database using Microsoft Access; constructing GIS file from textual data, building spatial queries, and measuring geographic distances for spatial database using PostgreSQL 9.4. By implementing spatial database and executing spatial queries, sophisticated geographic relationships among the studied objects were identified to help the understanding of geographic relationships.

# References

ArcGIS Online, 2014. World Countries. Accessed Apr 20, 2015, <http://www.arcgis.com/home/item.html?id=3864c63872d84aec91933618e3815dd2>.

Chapple M. Microsoft Access Sample Database: Countries, Cities and Provinces. Accessed Feb 2, 2015, <http://databases.about.com/od/sampleaccessdatabases/a/Microsoft-Access-Sample-Database-Countries-Cities-And-Provinces.htm>.

Kar, B. 2015. Entity-relationship model. GHY 567 (Design and Deployment of Databases and Geo-databases) lecture. Hattiesburg, MS: Department of Geography and Geology, The University of Southern Mississippi, Spring Semester.

PostGIS, 2015. Accessed Apr 20, 2015, <http://postgis.net/>.

QGIS, 2015. Accessed Apr 20, 2015, <http://www.qgis.org/en/site/>.

# Appendix

1. Select capital cities:

SELECT DISTINCT co.name, co.capital,city.geom

FROM project.country AS co, project.city

WHERE city.cityname = co.capital

2. Copy province.csv file:

COPY project.province FROM 'C:\Xiaohui\GeoDB\project\province.csv' WITH (FORMAT csv, HEADER True, NULL 'NULL');

3. Copy country\_privince file:

COPY project.con\_pro FROM 'C:\Xiaohui\GeoDB\project\country\_province.csv' WITH (FORMAT csv, HEADER True, NULL 'NULL');

4. Show the lat/lon of city:

SELECT city, ST\_AsText(geom)

FROM project.city;

5. Get the geom of country's captial and country's centroid

SELECT DISTINCT co.name, co.capital,city.geom As cap\_geom,ST\_Centroid(co.geom) As Centroid\_geom

FROM project.country AS co, project.city

WHERE city.cityname = co.capital

6. Get the distance btw captial and country's centroid in degree

SELECT name, capital,ST\_Distance(ST\_Transform(vw.cap\_geom,4326),

ST\_Transform(vw.centroid\_geom,4326)) As Dis\_Capi\_Centroid

FROM project.cap\_centri\_vw As vw

7. Get the max,min,avg of the distance calculated from step 6:

SELECT MAX(dis\_capi\_centroid),MIN(dis\_capi\_centroid),AVG(dis\_capi\_centroid)

FROM project.distance\_cap\_centroid\_vw