Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2009) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

World Conference on Innovation and Software Development

Materialization of BTA database using open source software

Jose Martinez-Llario[a], Francisco Ruiz-Lopez[a]*, Eloina Coll[a]

[a]*Instituto de Restauración del Patrimonio, Universitat Politècnica de València, Camino de Vera s/n. 46022 Valencia, Spain*

Abstract

Along this article we are going to explain how to create a new spatial database applying a wide spectrum data model using open source software and free tools, so that every interested user will be able to apply BTA data model completely cost free. This data model is ready to store cartographical information for scales similar to 1:5.000 or 1:10.000. We are going to use specific software that allows to control almost every aspect of this data model, both features represented and quality constraints.

Keywords: BTA; data model; cartography; open source; phenomena; JASPA; Kosmo; PostgreSQL; free.

1. Introduction

Some years ago the Specialized Committee of Cartographic Norms started the development of one new data model with very wide application field, the BTA[1].

This data model is designed to store cartographic data at scales between 1:5.000 and 1:10.000, although it can be used with similar scales.

It is also designed to store topographic cartography of almost every region of Spain; it means there many potential users, like Public Administrations or enterprises.

Moreover, the mentioned data model stores extra alphanumerical attributes of those geometries, becoming an interesting starting point to set up a powerful Geographical Information System at free cost.

Furthermore, this paper is an outline about how to do it with free and open source software. So increases considerably the potential market. The costs to set up this database ready to do spatial analysis with cartographic datasets highly interoperable are just the derived of the specialists technicians who manage it, not licenses or things like that.

For further information about this topic with commercial software please check [2] and [3].

2. Software used

JASPA[4] just released its 0.1 version last year, and PostGis has an extended experience under development and is now in its release 1.5.2, however along this paper we are going to work with JASPA by some interesting points. This software is written in JAVA, manages relational databases, brings spatial functions to help us with advanced

spatial analysis and the developer team works in the same investigation cluster.

Although there are other solid options available like PostGis we will choose JASPA because gives to us more freedom to develop tools to satisfy our own needs using Java.

Developing tools for PostGis environment is much more difficult to us because our team is composed mainly by cartographers and geodesy engineers, so our knowledge about programming is not strong enough to face an advanced PostGis customization.

For the visualization issues we are going to use Kosmo which is another open source software and it has proved working properly with those data bases.

As before, there are other options available, like Quantum GIS, however we consider Kosmo 2.0 the best option nowadays.

2.1. Remarkable properties of the software

As we can see in the attached image (Fig.1), Jaspa is developed following the directions of the Open Geospatial Consortium [5], and International Standard Organization. This is a guarantee of interoperability and standardization.

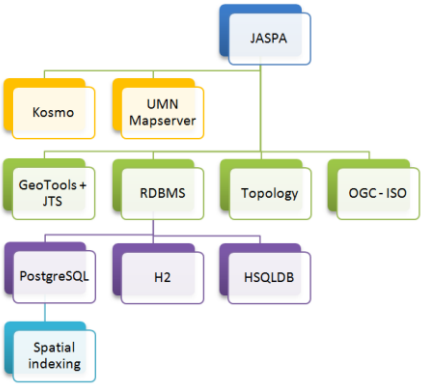

Fig. 1. Schematic JASPA components.

Jaspa, unlike PostGis, is ready to work with different relational databases, like PostgreSQL[6], H2DB[7] and the support for HSQLDB[8] is being implemented actually.

Also uses GeoTools for projections, working with reference systems, KML management, etc. and JTS[9] (Java Topology Suite), the API which allows the use of spatial analysis to our data.

Jaspa implements some topology rules too but is not fully supported yet, this face is under development although is expected in the future.

The powerful map server developed by the University of Minnesota, UMN Mapserver is used as a graphical frontend.

The spatial indexing is a very powerful and interesting characteristic of Jaspa, although it is only available with PostgreSQL databases.

3. Creation of the database

3.1. First steps

First of all we have to remark that all the controls and instructions made to the database are done with SQL sentences on the shell.

Secondly, we are going to create a new data base to store all the information. The sentence should be like “createdb -U username databasename”, and then, after that, we have to add the spatial Jaspa functionalities with

"createdb -U username -T jaspatemplate databasename".

Once these steps are done we can enter to the database to start the real work.

In our case we decided to use PostgreSQL as relational database manager so we will have the advantage of spatial indexing after the creation of the layers. Of course we have to create indexes and initialize them, if not it will work like if we would not done it.

3.2. Database structure

The figure 2 shows different elements available inside the database.

Domains, constraints and indexes are available for the entire database, and are common elements which can be used by the tables, spatial tables and views. Within a schema there are views and standard and spatial tables.

3.3. Schemas definition

Due to the complexity of this data model we are going to create schemas to simulate the families of the BTA phenomena catalogue.

The main reason for creating those schemas is to maintain the tables ordered.

A PostgreSQL schema is an aggregation of tables for representing common features. The data model considers eight families.

3.4. Domains definition

Once this step is finished we are not going to create the tables, because although we could do that and after that add the domains to the tables is much better create the domains before and use them at the moment of the table creation.

To create a domain we will use the following sentence "create domain domainname as datatype [default expression] [restriction1, …, restrictionk]".

If we take a look on the phenomena catalogue of the BTA data model we will see the large amount of domains particular for each phenomenon in the catalogue.

There are two kinds of domains, general domains, which have only one definition and are common for every phenomenon (like language domain) or particular domains, that could have the same name between phenomena but do not consider the same allowed values (like field kind for a waste depot or for a lightweight construction).

Because of that we have to distinguish between those domains for fields with the same name but which belong to different phenomena.

We suggest using the same name for the domain that the field name. For instance, "Waste depot" has phenomenon code 0050 inside the BTA, and the kind field name is "Tipo", so we recommend create the domain named "0050_Tipo".

Using the phenomenon's code before the field name we order implicitly the domains to maintain the database as much ordered as possible.

The order is extremely important working with BTA because there are more than 200 domains to create, if we do not do that the database will be chaotic.

Of course we can modify or erase existent domains in the database in case of need, look up the documentation website for more information.

3.5. Constraints definition

As a consequence of working with those RDBMS we are able to establish different constraints to guarantee some relations between elements inside the database. The manager system of the database is the responsible of the maintenance and force to follow the rules.

It is advisable to create constraints with clear names to avoid mistaken uses. Typical constraints are primary key, check, not null, etc. Using them we are guaranteeing the quality in the database.

The constraints used to be the same for all the fields of a family, even for the database, like the constraints for the

geometry field in the spatial tables.

The same way than happens with domains, we can alter tables adding constraints after the creation, but due to the large amount of tables we sincerely recommend to create it first, analyzing the requirements of the future spatial tables and prepare them before, to avoid mistakes.

To create a constraint we will use the following sentence, where the name is optional "[constraint constraintname] {domain-constraint | column-constraint | table-constraint}".

3.6. Tables definition

Once the domains, the schemas and the constraints are created, we are ready to start creating the spatial tables which represent the phenomena of the catalogue.

We will create spatial tables inside the schemas, and following a notation as much mechanic and clear as possible to be able to identify the layer easily once the database is ready.

There are two possible ways to do this task. The first is based in the capability of storing multiple geometry columns, so for each phenomenon we will create only one table, with as many geometry columns as possible representations has the phenomenon.

For instance the phenomenon "Road", at the table 1 we can see the fields created and the name of this spatial table "0027_Road". Road has polygonal and lineal representation, so it will need both geometry columns "Geometry1D" and "Geometry2D". The remaining fields are common for all those elements, so they appear only once.

We have to be cautious, because the 1D elements should have their own attribute values, that might differ from the superficial geometries. In this case we have to fill the geometry column with "No applicable" value in order to maintain the consistency.

Table 1. First option to create spatial tables for phenomena. Consider all the geometries together in one table.

Table	0027_Road	Domain
GEOMETRY2D	geometry	-
LOCATION	geometry	-
STATUS	Text	YES
TITULARITY	Text	YES
COMPETENCES	Text	YES
TRAMKIND	Text	YES
FUNCTIONALCLASSIF	Text	YES
LANGUAGE	Text	YES
NAME	Text	NO
ROADCODE	Text	NO

The other option is to split the geometries in tables depending on its dimension.

We recommend use the phenomenon code followed by the dimension of the geometry contained in the table.

From our experience we recommend do the second way, because although we have much more tables and the names are longer, nowadays there are not too many GIS software ready to work with tables which have more than one geometry column. And the existent software usually crashes so until powerful software appears, we will split the tables in with that methodology.

3.7. Defining indexes

To enhance the performance of the database it is highly recommendable to create spatial indexes, because when the amount of data stored and managed becomes big the performance of the database system decays strongly.

Creating indexes is a very easy task and it should be done once, because the database system cares for its

maintenance.

To create an index we use the following SQL sentence: “create index indexname on tablename using Gist (Stpgbox(geometrycolumn));”.

The same way than before, it is recommendable to associate the spatial index with the involved phenomenon table code.

4. Conclusion

Once the database is ready we have the BTA data model complete but empty, this is the moment to create a backup and share it with other interested people because all this work can be done once and reuse the database always.

The best way to export the database is using SQL files.

Finally the remaining task is to fill the data model with the geometries and the database is complete and ready to use.

Jaspa has demonstrated a good performance and flexibility with the required tasks. There is no appreciable difference between this database and commercial GIS databases.

Jaspa with PostgreSQL is ready to guarantee the quality standards, proceedings and requirements that BTA data model demands. Moreover it allows the customization, showing itself as a powerful tool ready to work and cost free.

SQL provides extra functionalities and customization possibilities that no other GIS commercial software available could achieve.

Acknowledgement

This project is a part of the research project “MOCAIDE”, Creation and cartographic feeding of spatial data infrastructures at the Local Administration trough a data model integrating cadastre, urban planning and historic heritage, with reference CSO2008-04808 and financed by the CICYT and European funds.

References

1. BTA Data model website, http://www.csg-cnc.es/web/cnccontent/bta.html
2. Ruiz, F., Coll, E.: Producción de Cartografía 1:5.000 ICV en formato BTA. Universidad Politécnica de Valencia, Valencia (2010)
3. Ruiz, F., Coll, E., Martinez-Llario, J.: Análisis y estructura de una Geodatabase BTA para su utilización en la Administración Local. JIDEE 2010 - Jornadas Ibéricas de la Infraestructura de Datos Espaciales, Lisboa (2010), http://www.idee.es/resources/presentaciones/JIIDE10/ID414_Analisis_y_estructura_de_una_Geodatabase_BTA.pdf
4. JASPA, http://forge.osor.eu/projects/jaspa/
5. OGC Website, http://www.opengeospatial.org/
6. PostgreSQL, http://www.postgresql.org/
7. H2DB Website http://www.h2database.com/html/main.html
8. HyperSQL Java Database, http://hsqldb.org/
9. Java Topology Suite, http://www.vividsolutions.com/jts/JTSHome.htm