Building a Spatial Database for Romanian Archaeological Sites

Aura-Mihaela MOCANU, Manole VELICANU
Economic Informatics Department, Academy of Economic Studies
Bucharest, ROMANIA
mocanuaura@yahoo.com, mvelicanu@yahoo.com

Spatial databases are a new technology in the database systems which allow storing, retrieving and maintaining geospatial data. This paper describes the steps which we have followed to model, design and develop a spatial database for Romanian archaeological sites and their assemblies. The system analysis was made using the well known Entity-Relationship model; the system design included the conceptual, the external and the internal schemas design, and the system development meant developing the needed database objects and programs. The designed database allows users to load vector geospatial data about the archaeological sites in two distinct spatial reference systems WGS84 and STEREO70, temporal data about the historical periods and cultures, other descriptive data and documents as references to the archaeological objects.

Keywords: *spatial databases, entity-relationship model, conceptual schema, external schema*

1 Introduction

Spatial databases have the ability of storing and manipulating geospatial data. They are usually extensions of relational databases which contain special geometry objects with several mandatory attributes and methods defined in the Open Geospatial Consortium's (OGC) standard "Application objects", spatial indexing mechanism, operators and functions to make queries, joins and other spatial analysis operations.

Geospatial data means raster data (Earth photography's made either from satellite, either from plane) or vector data (points, lines, polygons which describe the location of the objects on the surface of the Earth and their outline).

Some spatial available database systems are: commercial software Oracle Spatial or IBM DB2 Spatial Extender, the open source spatial databases such as: PostgreSQL/PostGIS, Spatial Box, Ingress Geospatial, H2 Spatial, Spatial Lite, MySQL Spatial. Spatial databases store the geospatial data by providing either a proprietary object such as Oracle's object SDO_GEOMETRY, either assuring the support for the standard storage format well-known binary (WKB). Paper [1] describes a detailed comparative study between Oracle Spatial and PostgreSQL/PostGIS.

Store the geospatial data is not enough; one must retrieve the data also, and fast. That is why the spatially-enabled databases also have defined a special type of indexes, named RTree indexes, which we have noticed to be available in most of the spatial databases which we have analyzed, both commercial and open source.

Spatial operations such as spatial queries, create, update, insert, and delete operations, conversions, and operations on the map or analysis on grid cells are very well documented in paper [2].

We will further describe the steps we have followed in order to build a spatial database for our collaborators from the Romanian National History Museum (MNIR) to manage the archaeological sites and their assemblies.

2 System analyses - Entity-Relationship Model

We have started the analysis of the new database by finding out what kind of data is it needed to be maintained by the specialists about the archaeological sites and their assemblies, having as example an old application which was used by them, written in Microsoft Access.

Like in any other archaeology-related information system, the data model includes the following data categories [3]:

- temporal data - historical periods and cultures are assigned to any archaeological site,
- spatial data - location of the archaeological site (descriptive and vector geospatial data),
- archaeological objects – description of archaeological sites and assemblies,
- documents – attached to any site.

The most used technique to create the data model is the Entity-Relationship (ER). In order to create the ER diagram, we have identified first the entities (users, archaeological sites, assemblies, geographical coordinates, waters, counties, historical periods, historical cultures, assemblies' classes) and their attributes. The main entities and their attributes are described in below table.

Table 1. Main entities and attributes

Entity	Attribute	Description
site	id_sit	ID site
	cod_siruta	SIRUTA code
	cod_lmi	LMI code
	cod_ran	RAN code
	denumire	name of the site
	alternativ	another site's name
	ape	waters
	punct	a point mark
	repere	landmarks
	sursa	site's references
	observatii	observations
ensemble	id_ansamblu	ID ensemble
	denumire	name
	clasa	ensemble's class
	tip	ensemble's type
	cercetare	research
	reper_supl	landmark
	inventar	inventory
	perioada	historical period
	cultura	historical culture
	sursa	references
	observatii	observations
coordinate	id_coordonate	ID coordinate
	tip_date	data type (information about GPS receiver)
	elevat	elevation
	eroare	error
	geom	geographical coordinates
	observatii	observations

After describing all the entities and their attributes, we have drawn the ER diagram (figure 1) with WebRatio tool, commercial software which supports the Web Modeling Language (WebML), whose purpose is the design of data intensive web sites. We have chosen this CASE tool because after building the database, a geoportal will be also designed.

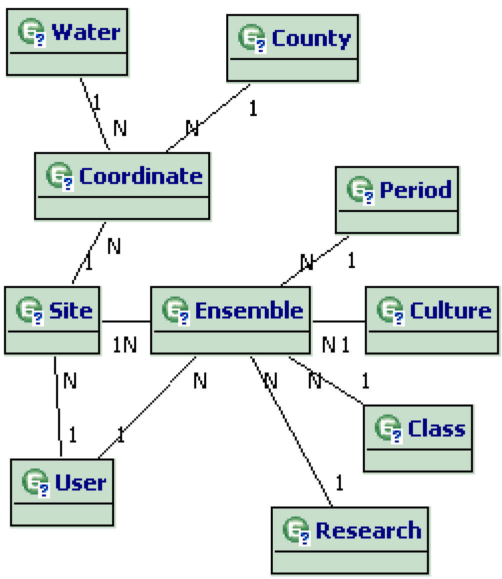

Fig. 1. ER diagram

The diagram has to be read as follows: one archaeological site is composed from one or more assemblies, which are loaded in the database by the system's users; every site has some geographical coordinates registered; the geographical coordinates refer not only to archaeological sites, but also to waters or Romania's counties boundaries; the archaeological assemblies dates from a certain historical period and culture, they belong to a certain ensemble class and they are part of a certain research program.

The relationships between the entities are described in below table:

Table 2. Relationships between entities

Entity	Linkage Phrase	One/ Many	Entity
Site	is composed from	many	Ensemble
Site	has assigned	many	Coordinate
Site	is last maintained by	one	User
Coordinate	refer to	one	Site
Coordinate	refer to	one	Water
Coordinate	refer to	one	County
Ensemble	is part of	one	Site
Ensemble	is last maintained by	one	User
Ensemble	dates from	one	Period
Ensemble	belongs to	one	Culture
Ensemble	belongs to	one	Class
Ensemble	is part of	one	Research
Period	belongs to	many	Ensemble
Culture	belongs to	many	Ensemble
Research	studies	many	Ensemble
Class	describes	many	Ensemble
User	maintain	many	Site
User	maintain	many	Ensemble

The "geom" attribute of "coordinate" entity is the vector geospatial data: points, lines or polygons representing the archaeological sites in two different spatial reference systems (SRS): WGS84 (which is the world's most used SRS) or STEREO 70 (which is Romania's most used SRS).

The geospatial data describing the Romania's counties and waters is loaded from a shapefile (.shp) provided by the Romanian geospatial community (http://earth.unibuc.ro/). The shapefiles are the ESRI's proprietary storage format for geospatial data.

3 System design

PostgreSQL/PostGIS was chosen to be used as the spatially-enabled database by analyzing the following decision factors:

- technical capabilities: it includes support for all of the functions and objects defined in the OpenGIS 'Simple Features for SQL' specification,
- documentation: it is available online with a lot of coding examples,
- support: it has an online bug tracking mechanism (http://trac.osgeo.org),
- ease of use: it is easy to install, easy to develop the database,
- usage rate: many scholars have used with success this database for their projects,
- price: open source tools are very interesting for the archaeologists who usually deal with low budget projects.

3.1 Conceptual Schema

The conceptual schema of the database is designed starting from the previous modeled ER diagram. Each entity from the ER model is transformed into a database table, and for each relationship foreign keys (FK) are defined. The conceptual schema will be improved step by step, following the normalization technique, until a balance is reached between the maintenance requirements and system's exploitation performance [4].

The corresponding database tables for the main entities which we have described in the previous section are detailed in below table.

Table 3. SITE table description

Column description	Column name	Data Type	PK	FK
ID site	id_sit	integer	X	
SIRUTA Code	cod_siruta	integer		X
LMI Code	cod_lmi	integer		X
RAN Code	cod_ran	integer		X
Name	denumire	text		
Alternative name	alternativ	text		
Point	punct	text		
Landmarks	repere	text		
References	sursa	text		
Comments	observatii	text		

Table 4. ASSEMBLIES table description

Column description	Column name	Data Type	PK	FK
ID site	id_sit	integer		X
ID ensemble	id_ansamblu	integer	X	
Name	denumire	text		
ID assemblies class	id_clasa	integer		X
ID assemblies types	id_tip	integer		X
ID research	id_cercetare	integer		X
Landmark	reper_supl	text		
Inventory	inventar	text		
ID historical period	id_perioada	integer		X
ID historical culture	id_cultura	integer		X
References	sursa	text		
Comments	observatii	text		
Last mutation date	data_ultima_actualiz	timestamp		
Last mutation user	utiliz_ultima_actualiz	text		

Table 5. COORDINATES table description

Column description	Column name	Data Type	PK	FK
ID coordinate	id_coordonate	integer	X	
ID sit	id_sit	integer		X
Data type (GPS receiver)	tip_date	text		
Elevation	elevat	integer		
Error	eroare	numeric		
Geographical coordinates	geom	geometry		
Comments	observatii	text		

Because "geom" column of "coordinates" table is defined as a geometry data type column (being used to store vector geospatial data about the archaeological sites), it has to be included in special PostGIS table "geometry_columns". This table defines: all the tables containing geometry columns, the spatial dimension (2, 3 or 4 dimensional) of the geometry columns, the ID of the spatial reference system used for the coordinate geometry, and the type of the spatial object (POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION).

The spatial reference systems which could be assigned to the geospatial data stored in PostGIS are defined in PostGIS table named "spatial_ref_sys". This technical table lists over 3000 known spatial reference systems (SRS) and details needed to transform/reproject between them. The geospatial data which will be filled by the users of currently described system might be in the following spatial reference systems: WGS84 whose corresponding PostGIS SRID is 4326 and STEREO70 whose SRID is 31700. One can find out the Well-Known Text (WKT) representation of a certain SRS by using the following select statement:

```
select srtext from spatial_ref_sys where srid = 4326;

"GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],TOWGS84[0,0,0,0,0,0,0],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]"
```

The WKT representation of a spatial reference system offers a standard to describe, as a text, the information about the geospatial data projection system. The projection system must be specified in the geospatial data source (file or database), because it is very important especially when the data from different sources is used together. One can overlay on a map only layers in the same projection (or spatial reference system). The WGS84 (World Geodetic System of 1984) models the world as a spheroid. No ellipsoid/spheroid model perfectly the Earth, but corrections are made in order to get a better approximation for each territory. These special corrections of the initially spheroid are named datum [5]. In our country the Stereographic 70 (STEREO70) system was adopted.

3.2 External Schema

In order to draw maps in different spatial reference system, we have created several views (table 6) on the database table COORDINATES filtering the geospatial data by its type (point, line or polygon) and by the spatial reference system in which it was represented.

Table 6. Database views

View	Description
vw_point_wgs	The representation of the archaeological sites, as points, in 4326 projection (also known as WGS84).
vw_point_stereo	The representation of the archaeological sites, as points, in 31700 projection (also known as Stereo70).
vw_point_google	The representation of the archaeological sites, as points, in 900913 projection (also known as Web Mercator).
vw_polygon_wgs	The representation of the archaeological sites, as polygons, in 4326 projection (also known as WGS84).
vw_polygon_stereo	The representation of the archaeological sites, as polygons, in 31700 projection (also known as Stereo70).
vw_polygon_google	The representation of the archaeological sites, as polygons, in 900913 projection (also known as Web Mercator).

These views are important when the map layers will be defined in the Web Mapping Service (WMS) software, because one layer can be associated with a “geometry” database column only if this column contains data about one single type of geospatial data, in one single SRS. We will detail in section 4 of this paper how the geospatial data stored in the database is rendered on a map by using WMS software.

Regarding the security policy, there were designed two user group roles with different rights in the database. The select action on the tables is granted to user role „guest” and insert/update/delete actions are granted to the user role „admin”. Each time a user is created, one of the two group roles („guest” or „admin”) will be granted to the user, so that the access to the database is done in a rigorous way. Also for security reasons, each change on the 3 main tables (archaeological sites, assemblies and coordinates) will trigger an insert action into a history table, so that the „admin" role user will know when was it done an update on these tables, by which user and what data did he modified.

3.3 Internal Schema

In this step we have defined RTree indexes for each table which contains geometry columns. RTree approximates each geometry through a *Minimum Bounding Rectangle (MBR)*. The Postgres query optimizer will consider using an RTree index whenever an indexed attribute is involved in a comparison using one of the following geometric operators: <<, &<, &>, >>, @, ~=, && which are explained in below table:

Table 7. Geometric operators

Operator	Description
&&	Overlaps?
~=	Same as?
@	Center
<<	Is strictly left of?
>>	Is strictly right of?
&<	Does not extend to the right of?
&>	Does not extend to the left of?

The disk space can be estimated in three ways: using special SQL functions (table 8), using **`VACUUM`** information, and from the command line using the tools in **`contrib/oid2name`**. The SQL functions are the easiest to use and report information about tables, tables with indexes and long value storage (TOAST), databases, and tablespaces. Using psql on a recently vacuumed or analyzed database, queries could be written to see the disk usage of any table [6], such as:

```
SELECT relfilenode, relpages FROM
pg_class WHERE relname = 'coordonate';
```

Each page is typically 8 kilobytes and the **`relfilenode`** value is of interest to examine the table's disk file directly.

Table 8. Database Object Size Functions

Name	Return Type
`pg_column_size(any)`	`int`
`pg_database_size(oid)`	`bigint`
`pg_database_size(name)`	`bigint`
`pg_relation_size(oid)`	`bigint`
`pg_relation_size(text)`	`bigint`
`pg_size_pretty(bigint)`	`text`
`pg_tablespace_size(oid)`	`bigint`
`pg_tablespace_size(name)`	`bigint`
`pg_total_relation_size(oid)`	`bigint`
`pg_total_relation_size(text)`	`bigint`

4 System developments

We have first created the database tables, as shown in figure 2.

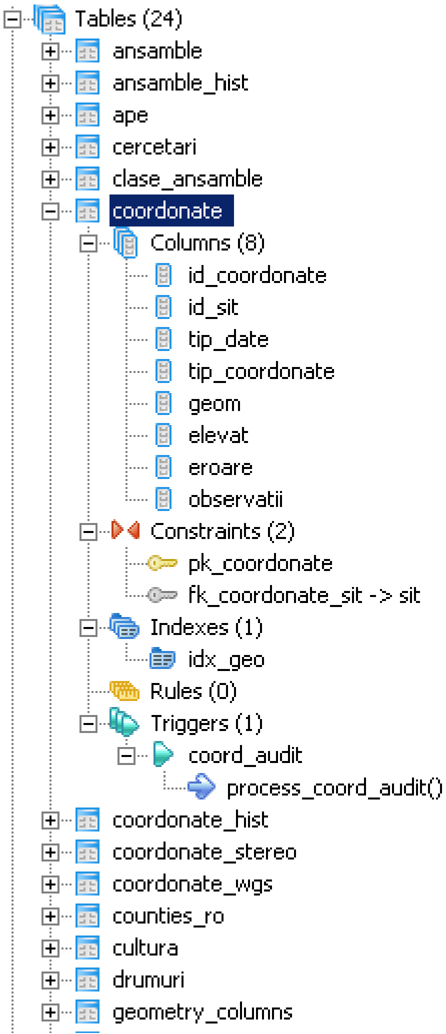

Fig. 2. Database tables

In PostGIS, there is used following syntax in order to add a geospatial column:
AddGeometryColumn(<table_name>, <column_name>, <srid>, <data_type>, <dimension number>)

For example, we have used the following statement in order to add "geom" column from "coordonate" table:

```
AddGeometryColumn(coordonate, geom,
4326, point, 2);
```

The PostGIS storage format for the geometric object is Well Known Binary (WKB). The Well Known Binary Representation for Geometry (WKBGeometry) provides a portable representation of a geometric object as a contiguous stream of bytes. It permits geometric object to be exchanged between an SQL/CLI client and an SQL-implementation in binary form. The Well-known Binary Representation for Geometry is obtained by serializing a geometric object as a sequence of numeric types drawn from the set {Unsigned Integer, Double} and then serializing each numeric type as a sequence of bytes using one of two well defined, standard, binary representations for numeric types (NDR, XDR) [7].

Table 9. Integer codes for geometric types

Type	Code
Geometry	0
Point	1
LineString	2
Polygon	3
MultiPoint	4
MultiLineString	5
MultiPolygon	6
GeometryCollection	7
CircularString	8
CompoundCurve	9
CurvePolygon	10
MultiCurve	11
MultiSurface	12
Curve	13
Surface	14
PolyhedralSurface	15
TIN	16

The master data tables which store the geospatial data regarding the Romanian counties and waters were loaded from an ESRI shapefile, into the database, using the *shp2pgsql* tool according to the below schema.

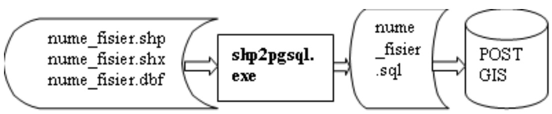

Fig. 3. Load geospatial data from a shapefile into the database

A GiST index was created on "coordonate" table:

```
CREATE INDEX idx_geo ON coordonate
 USING gist (geom);
```

GiST stands for Generalized Search Tree. It is a balanced, tree-structured access method that acts as a base template in which to implement arbitrary indexing schemes. B-trees, R-trees and many other indexing schemes can be implemented in GiST [6].

The views on “coordonate” table, used to split geospatial data by its type and SRS, were created using some PostGIS spatial functions, such as ST_TRANSFORM. For example, to create a view with all the POINT geospatial data, either in WGS84, either in STEREO70, and reprojected in Web Mercator SRS, we have used below statement:

```
CREATE OR REPLACE VIEW vw_point_google
AS
SELECT coordonate.id_sit,
sit.denumire, sit.cod_siruta,
sit.repere, sit.ape,
coordonate.tip_date,
st_transform(coordonate.geom, 900913)
AS geom, coordonate.tip_coordonate,
coordonate.elevat, coordonate.eroare,
coordonate.observatii
FROM coordonate, sit
WHERE coordonate.id_sit = sit.id_sit
AND  geometrytype(coordonate.geom)  =
'POINT'::text;
```

PostGIS function ST_TRANSFORM uses the open source PROJ4 library in order to reproject the geospatial data from one spatial reference system to another. PROJ4 is a cartographic projections library used by many other GIS tools to reproject the geospatial data from one spatial reference system (SRS) to another. This function returns a new geometry with its coordinates transformed to spatial reference system referenced by the SRID integer parameter. The destination SRID must exist in the `“spatial_ref_sys”` table.

Some history tables of the main tables (“sit”, “ansamble”, and “coordonate”) were created. These history tables will be filled with the old data from their corresponding tables when each maintenance action will happen. This automatically action is possible by developing some triggers.

The trigger will be associated with the specified table and will execute the specified function *funcname* when certain events occur. The trigger can be specified to fire either before the operation is attempted on a row (before constraints are checked and the **INSERT**, **UPDATE**, or **DELETE** is attempted) or after the operation has completed (after constraints are checked and the **INSERT**, **UPDATE**, or **DELETE** has completed). If the trigger fires before the event, the trigger may skip the operation for the current row, or change the row being inserted (for `INSERT` and `UPDATE` operations only). If the trigger fires after the event, all changes, including the last insertion, update, or deletion, are “visible” to the trigger [6].

For example, the trigger which was developed for “coordonate” table:

```
CREATE TRIGGER coord_audit
  AFTER INSERT OR UPDATE OR DELETE
  ON coordonate
  FOR EACH ROW
  EXECUTE PROCEDURE
process_coord_audit();

CREATE OR REPLACE FUNCTION
process_coord_audit()
  RETURNS trigger AS
$BODY$
  BEGIN
    IF (TG_OP = 'DELETE') THEN
       INSERT INTO coordonate_hist
SELECT OLD.*;
       RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
       INSERT INTO coordonate_hist
SELECT OLD.*;
       RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
       INSERT INTO coordonate_hist
SELECT NEW.*;
       RETURN NEW;
    END IF;
  RETURN NULL;
  END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
ALTER FUNCTION process_coord_audit()
OWNER TO postgres;
```

PL/pgSQL can be used to define trigger procedures. A trigger procedure is created with the **CREATE FUNCTION** command, declaring it as a function with no arguments and a return type of `trigger`. The function must be declared with no arguments even if it expects to receive arguments specified in **CREATE**

TRIGGER - trigger arguments are passed via TG_ARGV.

In order to render the geospatial data about the archaeoloical sites on a map (figure 4), we have further configured the open source WMS **GeoServer**. The WMS has as output maps of spatially referenced data dynamically from geographic information. To make the interoperability possible between GeoServer and PostGIS, we have defined in GeoServer a connection to PostGIS database („data store”) by providing the database name, location, port; the layers of the map („feature types”) and the style in which the map will be drawn. In GeoServer, a feature type was created for each of the designed views containing the geospatial data. The style means how the layers will be drawn on the map: which colors and which symbols to use for our point or linestring or polygon data. To define a style, a XML based file named Style Layer Description (SLD) file must be developed. SLDs were developed for all the map layers, defining that the counties bounderies are drawn with black lines, the waters with blue lines and the archaeological sites with red dots (in case of POINT data) or red lines (in case of LINESTRING/POLYGON data).

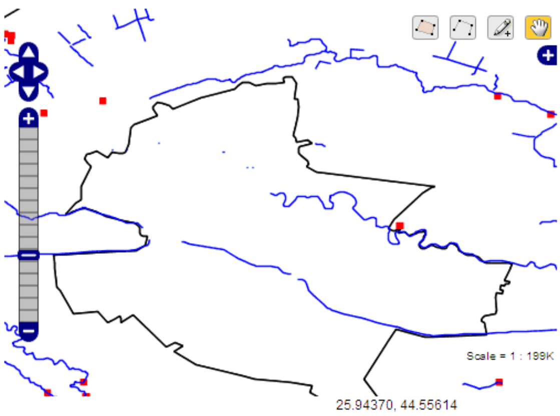

Fig. 4. Map rendered by GeoServer for the geospatial data stored in PostGIS

5 Conclusions

In this paper, we have shown that the spatial databases are modeled and designed like any other database, drawing the ER diagram, defining the conceptual, external and internal schemas, taking care though by one very important particularity: the “geometry” type column of some tables which store geospatial data as well. In order to use this kind of column further, when integrating the spatial database with a Web Mapping Server (WMS), the geospatial data has to be split by its type (point/linestring /polygon) and by its spatial reference system (SRS). Our proposal of creating different views for each type / SRS combination has proved to be an easy and efficient way for making possible the interoperability between PostGIS database and GeoServer WMS. Also, we have shown that the open source database PostGIS can successfully be used to develop a rigorous spatial database.

References

[1] D. Litan, A.M. Mocanu, S. Olaru, A. Apostu. “Modern Information Technologies Used In Market Research”. *Proceedings of the 9th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS’10)*, pp. 245-250, Merida, Venezuela, December 14-16, 2010

[2] A. Velicanu. “Spatial Operations”. *Database Systems Journal*, vol. 1, pp. 5-8, 2010

[3] E. Meyer, P.Grussenmeyer, J.P. Perrin, A. Durand and P.Drap. „A web information system for the management and the dissemination of Cultural Heritage data”. *Journal of Cultural Heritage*, vol. 8 (4), pp. 396-411, 2010

[4] M. Velicanu, M. Muntean, I. Lungu, S. Ionescu, *Sisteme de baze de date*, Ed. Petrion, Bucharest, 2003

[5] M. Bădut, *Sisteme geoinformatice pentru electroenergetică*, Ed. Polirom, Iasi, 2008

[6] PostgreSQL, *PostgreSQL 8.2.20 Documentation*, http://www.postgresql.org/docs/8.2/static/index.html

[7] Open Geospatial Consortium, *OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture*, http://www.opengeospatial.org/standards/sfa

Aura-Mihaela MOCANU has graduated The Bucharest Academy of Economic Studies, Faculty of Cybernetics, Statistics and Economic Informatics in 2007. She holds a Master diploma in Databases - Support for business from 2009 and in present she is a PhD Candidate in Economic Informatics with the Doctor's Degree Thesis: Software technologies to build a Geographical Information System for a public institution. Her areas of interest are: Geographical Information Systems, Databases, Information Systems integration, Programming languages.

Manole VELICANU is a Professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the Faculty of Economic Cybernetics in 1976, holds a PhD diploma in Economics from 1994 and starting with 2002 he is a PhD coordinator in the field of Economic Informatics. He is the author of 18 books in the domain of economic informatics, 64 published articles (among which 2 articles ISI indexed), 55 scientific papers published in conferences proceedings (among which 5 papers ISI indexed and 7 included in international databases) and 36 scientific papers presented at conferences, but unpublished. He participated (as director or as team member) in more than 40 research projects that have been financed from national research programs. He is a member of INFOREC professional association, a CNCSIS expert evaluator and a MCT expert evaluator for the program Cercetare de Excelenta - CEEX (from 2006). From 2005 he is co-manager of the master program Databases for Business Support. His fields of interest include: Databases, Design of Economic Information Systems, Database Management Systems, Artificial Intelligence, Programming languages.