Mysql spatial and postgis-implementations of spatial data standards

Article · January 2011

CITATIONS
17

READS
5,079

1 author:

Adam Piórkowski
AGH University of Science and Technology in Kraków
86 PUBLICATIONS **585** CITATIONS

SEE PROFILE

Electronic Journal of Polish Agricultural Universities (EJPAU) founded by all Polish Agriculture Universities presents original papers and review articles relevant to all aspects of agricultural sciences. It is target for persons working both in science and industry, regulatory agencies or teaching in agricultural sector. Covered by IFIS Publishing (Food Science and Technology Abstracts), ELSEVIER Science - Food Science and Technology Program, CAS USA (Chemical Abstracts), CABI Publishing UK and ALPSP (Association of Learned and Professional Society Publisher - full membership). Presented in the Master List of Thomson ISI.

ELECTRONIC JOURNAL OF POLISH AGRICULTURAL UNIVERSITIES

2011
Volume 14
Issue 1
Topic
GEODESY AND CARTOGRAPHY

PIÓRKOWSKI A., MYSQL SPATIAL AND POSTGIS – IMPLEMENTATIONS OF SPATIAL DATA STANDARDS, EJPAU, 14(1), #03.
Available Online http://www.ejpau.media.pl

MYSQL SPATIAL AND POSTGIS – IMPLEMENTATIONS OF SPATIAL DATA STANDARDS

Adam Piórkowski

Department of Geoinfomatics and Applied Computer Science,
AGH University of Science and Technology, Cracow, Poland

ABSTRACT

Spatial databases have become a very important domain of databases recently. A standardization of the storing and analyzing methods provides consistency of data, makes it easier to combine these data and allows to integrate projects.
There is a problem of standardization for leading public domain databases with spatial extensions described in this article. The two main standards - OGC OpenGIS and SQL/MM are presented. The consideration focuses primarily on two databases: PostgreSQL + PostGIS and MySQL Spatial. The problem of standard implementation is the main issue. The way of integration spatial extension with database system is described. The level of standard implementation is discussed. Examples for analytical functions are given. Several available language constructions of creating queries are presented.
The integration and implementation level of presented spatial extensions differs. MySQL Spatial is well integrated, but realizes only part of OGC OpenGIS standard. PostGIS implements two standards almost entirely, but there way of spatial database creating is much more complicated. The considerations are illustrated with examples in SQL.

Key words: spatial data, spatial databases, ogc, opengis, sql, sql/mm spatial, postgresql, postgis, mysql spatial

INTRODUCTION

The dynamic development of spatial information systems is a source of vast amounts of data. While it is possible to create implementations only dedicated to the processing of such data, the binding of these data with other information resources, often giant, is difficult. The solution is the construction of integrated systems with spatial information management systems using relational databases.

Spatial data in relational databases can be saved according to dedicated projects and needs, even by adding an additional columns with coordinates describing the facts gathered. This solution is not flexible, is tied to the project. Introducing the class custom implementations of spatial data is also not a good option, as it often is dependent on the used database product, it is not portable, and the lack of coupling with other solutions provide difficulties in exchanging or transferring data. These problems can be bypassed by introducing a standard that defines the basic types, operations and ways to store spatial data. There are two spatial data standards defined for relational databases: OpenGIS (OGC) [8,9] and SQL/MM [2,4].

OpenGIS is spatial data standard for relational databases, proposed by Open Geospatial Consortium. Standard SFS (Simple Features SQL) includes the objects, the format of the recording, basic operations on them, and indexing. Unfortunately, the problem is solved only for the development of the planar systems (2D). The downside is

unresolved by the end of the definition of reference systems. The standard of SQL / MM (Part Three – Spatial), which inter alia allows for more dimensions of data, description of objects as well as curves, better integration of systems of reference, and provides many more features. Described briefly the standards are implemented in varying levels in popular database management systems, including in the public domain.

Spatial extensions for relational databases

There are spatial data extensions for each relational database management system. In most cases these extensions are built in. The Oracle databases have own extension Oracle Spatial (Oracle Locator) [10,14,3]. IBM DB2 uses Spatial Extender to provide spatial data [1]. The version 2008 of MS SQL provides Spatial Storage – a module for spatial data [5]. SQLite uses SpatiaLite as a spatial extension [13].

MySQL uses MySQL Spatial extension [6]. MySQL Spatial is well integrated with database management system, but realizes only part of one standard (OpenGIS) and only the 2D dimension without reference sets. The spatial extension is available only for MyISAM, InnoDB, NDB, BerkeleyDB (obsolete, removed) and ARCHIVE database engines. MySQL is released under the General Public License (GPL) and the commercial license.

The standalone version of PostgreSQL [12] database doesn't provide a spatial extension, but there is an outer open solution – PostGIS [11,7]. PostGIS implements two standards almost entirely, but it has some limitations associated with installation and integration. It provides own extensions for spatial data – extended formats of spatial data (EWKT, EWKB, HEXEWKB), mode dimensions (2D, 3D, 3DZ, 3DM), and geography objects (using sphere or spheroid). PostGIS is released under the GNU General Public License [11].

SPATIAL DATA STANDARDS

There are two open standards of spatial extension for databases: OpenGIS and SQL/MM Spatial.
OpenGIS (SFS - Simple Features SQL) is a standard by the Open Geospatial Consortium. It provides objects definition, data format (WKT, WKB), set of spatial functions and methods of indexing. Weaknesses of OpenGIS standard are the limitations of object dimensions to 2D and the lack of spatial reference systems.

The SQL/MM Spatial standard provides much more functions (for example defines the objects approximated with curves) and enables more dimensions of objects.

Naming convention

OpenGIS is an earlier standard. The proposed names are too general and may overlap with other functions of the database. To avoid such a situation the newer standard SQL/MM Spatial provides names that begin with the prefix ST_. Standard SQL/MM Spatial implements all standard OpenGIS features. Parameters of function calls for both standards are identical. The function names are very similar - the comparison are summarized in Tables 1 (geometrical objects) and 2 (functions).

Table 1. Naming convention for objects

OpenGIS	SQL/MM
Point	ST_Point
Curve	ST_Curve
Linestring	ST_Linestring
	ST_Circularstring
	ST_CompoundCurve
Surface	ST_Surface
	ST_CurvePolygon
Polygon	ST_Polygon
GeomCollection	ST_Collection
Multipoint	ST_Multipoint
Multicurve	ST_MultiCurve
Multilinestring	ST_Multilinestring
Multisurface	ST_Multisurface
Multipolygon	ST_Multipolygon

Table 2. Naming convention for functions

OpenGIS	SQL/MM
Equals	ST_Equals
Disjoint	ST_Disjoint
Touches	ST_Touches
Within	ST_Within
Overlaps	ST_Overlaps
Crosses	ST_Crosses
Intersects	ST_Intersects
Contains	ST_Contains
Relate	ST_Relate

IMPLEMENTATIONS OF SPATIAL DATA STANDARDS IN MYSQL SPATIAL AND POSTGIS

Integration

MySQL Spatial is closely integrated with the database management system. This allows to easily create tables that contain columns of geometric objects. Geometric types are already available after installation. It is possible to add geometric columns to an existing database. The example below shows the creation of tables describing municipalities (area represented by POLYGON) and roads (mileage represented by LINESTRING).

```
mysql=> CREATE DATABASE daneprzestrzenne;
mysql=> CONNECT daneprzestrzenne;
mysql=> CREATE TABLE Gminy(id_gminy INT, nazwa_gminy VARCHAR(100), obszar POLYGON);
mysql=> CREATE TABLE Drogi(id_drogi INT, przebieg LINESTRING);
```

The same operation in PostGIS is hard to perform when database already exists or database was created without a proper template. The example below shows an error, that occurs in this case.

```
psql=# CREATE DATABASE przestrzenna1;
CREATE DATABASE

psql=# \c przestrzenna1
You are now connected to database "przestrzenna1".

psql=# CREATE TABLE Gminy(id_gminy INT, nazwa_gminy VARCHAR(100));
CREATE TABLE

psql=# SELECT AddGeometryColumn('', 'gminy','obszar',-1,'POLYGON',2);
ERROR:  function addgeometrycolumn(unknown, unknown, unknown, integer, unknown,
integer) does not exist at character 8
HINT:  No function matches the given name and argument types. You might need to add
explicit type casts.
```

To create database with geometrical objects the selection of proper template is needed. Geometrical columns can be added by special function *AddGeometryColumns* in the way shown on example below.

```
psql=# CREATE DATABASE przestrzenna TEMPLATE=template_postgis;
CREATE DATABASE

psql=# \c przestrzenna
You are now connected to database "przestrzenna".

psql=# CREATE TABLE Gminy(id_gminy INT, nazwa_gminy VARCHAR(100));
CREATE TABLE

psql=# SELECT AddGeometryColumn('', 'gminy','obszar',-1,'POLYGON',2);
                addgeometrycolumn
-------------------------------------------------
 public.gminy.obszar SRID:-1 TYPE:POLYGON DIMS:2
(1 row)
```

An alternate way to enable spatial features for existing database is to run a set of scripts written in plpgsql language.

Standard functions implementation

Reflections on the level of implementation of the standard of value is illustrated by the example. There are four roads and one municipality on a simple map (Fig.1).

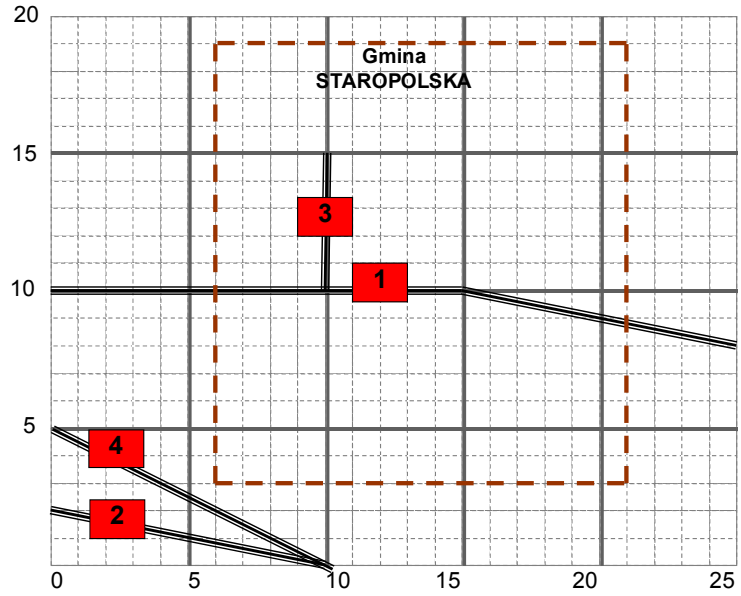

Fig. 1. A simple map

To describe these objects in database following commands are needed to be executed:

```
INSERT INTO Gminy (id_gminy, Nazwa_gminy, obszar)
 VALUES (1, 'Staropolska',
                GeomFromText('POLYGON((6 3, 6 19, 21 19, 21 3, 6 3))'));

INSERT INTO Drogi(id_drogi, przebieg)
 VALUES (1, GeomFromText('LINESTRING(0 10, 15 10, 25 13)', -1));
INSERT INTO Drogi(id_drogi, przebieg)
 VALUES (2, GeomFromText('LINESTRING(0 2, 10 0)', -1));
INSERT INTO Drogi(id_drogi, przebieg)
 VALUES (3, GeomFromText('LINESTRING(10 10, 10 15)', -1));
INSERT INTO Drogi(id_drogi, przebieg)
 VALUES (4, GeomFromText('LINESTRING(0 5, 10 0)', -1));
```

Simple functions

The example below shows, that simple functions like length calculating are correct for both database systems. In MySQL Spatial length calculating is performed by function *GLENGTH* (not defined by standard) because function *LENGTH* is reserved for strings.

```
mysql> SELECT id_drogi, GLENGTH(przebieg) FROM Drogi;
+----------+--------------------+
| id_drogi | GLENGTH(przebieg)  |
+----------+--------------------+
|        1 |  25.4403065089106  |
|        2 |  10.1980390271856  |
|        3 |                  5 |
|        4 |  11.1803398874989  |
+----------+--------------------+
```

```
mysql> SELECT id_drogi, LENGTH(przebieg) FROM Drogi;
+----------+------------------+
| id_drogi | LENGTH(przebieg) |
+----------+------------------+
|        1 |               61 |
|        2 |               45 |
|        3 |               45 |
|        4 |               45 |
+----------+------------------+
```

```
psql=# SELECT id_drogi, LENGTH(przebieg) FROM Drogi;
 id_drogi |      length
----------+------------------
        1 | 25.4403065089106
        2 | 10.1980390271856
        3 |                5
        4 | 11.1803398874989
(4 rows)
```

3.2.2 Relations between objects

In this section, there is a problem of differences in implementations of the standard described. This includes designating function relationships between objects. In the case of PostGIS simple functions of intersection and disjoint work correctly, in the case of MySQL Spatial - no. This is illustrated by the following example.

```
mysql> SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                    WHERE Contains(obszar, przebieg);
+----------+-------------+
| id_drogi | Nazwa_gminy |
+----------+-------------+
|        3 | Staropolska |
+----------+-------------+
1 row in set (0.00 sec)
```

```
psql=# SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                    WHERE Contains(obszar, przebieg);

 id_drogi | nazwa_gminy
----------+-------------
        3 | Staropolska
(1 row)
```

Containing is tested properly in both systems for this example. Wrong results occur for *Disjoint* and *Intersects* functions.

```
mysql> SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                   WHERE Disjoint(obszar, przebieg);
+----------+-------------+
| id_drogi | Nazwa_gminy |
+----------+-------------+
|        2 | Staropolska |
+----------+-------------+
1 row in set (0.00 sec)
```

```
psql=# SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                   WHERE Disjoint(obszar, przebieg);
 id_drogi | nazwa_gminy
----------+-------------
        2 | Staropolska
        4 | Staropolska
(2 rows)
```

In the presented example the Road number 4 is not classified by MySQL Spatial as disjoint from municipality. Intersection is not tested correctly too:

```
mysql> SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                    WHERE Intersects(obszar, przebieg);
+----------+-------------+
| id_drogi | Nazwa_gminy |
+----------+-------------+
|        1 | Staropolska |
|        3 | Staropolska |
|        4 | Staropolska |
+----------+-------------+
3 rows in set (0.00 sec)
```

```
psql=# SELECT id_drogi, Nazwa_gminy FROM Drogi, Gminy
                                    WHERE Intersects(obszar, przebieg);
 id_drogi | nazwa_gminy
----------+-------------
        1 | Staropolska
        3 | Staropolska
(2 rows)
```

MySQL currently does not implement this functionality properly. These functions are replaced by analogous functions that operate on the minimum bounding rectangles (MBR). For example, calling the intersection function returns the result of the intersection for MBR.

```
Intersects(@geom, @geoin)
MBRIntersects(@geom, @geoin)
```

Reduction of object dimensions

Another difference is an ability to reduce dimensions of objects, that are results of some functions. PostGIS can produce envelope that is 1D for object of such dimensions, MySQL Spatial produces polygons for all.

```
mysql> SELECT id_drogi, AsText(ENVELOPE(przebieg)),
                    Dimension(ENVELOPE(przebieg)) AS "Wymiar" FROM Drogi;
+----------+-----------------------------------------+--------+
| id_drogi | AsText(ENVELOPE(przebieg))              | Wymiar |
+----------+-----------------------------------------+--------+
|        1 | POLYGON((0 10,25 10,25 13,0 13,0 10))   |      2 |
|        2 | POLYGON((0 0,10 0,10 2,0 2,0 0))        |      2 |
|        3 | POLYGON((10 10,10 10,10 15,10 15,10 10)) |      2 |
|        4 | POLYGON((0 0,10 0,10 5,0 5,0 0))        |      2 |
+----------+-----------------------------------------+--------+
```

```
psql=# SELECT id_drogi, AsText(ENVELOPE(przebieg)),
           ST_Dimension(ENVELOPE(przebieg)) AS "Wymiar" FROM drogi;
 id_drogi |                astext                 | Wymiar
----------+---------------------------------------+--------
        1 | POLYGON((0 10,0 13,25 13,25 10,0 10)) |      2
        2 | POLYGON((0 0,0 2,10 2,10 0,0 0))      |      2
        3 | LINESTRING(10 10,10 15)               |      1
        4 | POLYGON((0 0,0 5,10 5,10 0,0 0))      |      2
```

Aggregation of data

The aggregation of data allows to create queries that process a set of object. There is a simple implementation of aggregation implemented in both database systems. For example calculating the sum of all road lengths can be processed in the way shown below:

```
mysql> SELECT SUM(GLength(przebieg)) FROM Drogi;
+-------------------------+
| SUM(GLength(przebieg)) |
+-------------------------+
|       51.8186854235951 |
+-------------------------+
1 row in set (0.00 sec)
```

```
psql=# SELECT SUM(ST_Length(przebieg)) FROM Drogi;
       sum
------------------
 51.8186854235951
(1 row)
```

PostGIS enables advanced query constructions, that contain arrays or unions of entities. MySQL Spatial has no implementation for such operations like these shown below:

```
psql=# SELECT Length(ST_Union(ARRAY(SELECT przebieg FROM drogi)));
      length
------------------
 51.8186854235951
(1 row)
```

```
psql=# SELECT Length(ST_Union(przebieg)) FROM drogi;
      length
------------------
 51.8186854235951
(1 row)
```

Spatial operators

Simple geometric operations with spatial operators are implemented only in PostGIS. MySQL has no implemented functions, that return an geometrical object as result. This applies to operators like *ConvexHull*, *Difference*, *Intersection* and *Union*.

CONCLUSION

Not only commercial database management systems, but also a public domain has extensions that allow the storage of spatial data. Portability and unification is ensured by the public standards: OGC SFS and SQL/MM Spatial. The level of integration and implementation of expansion is different – MySQL Spatial is well integrated, but realizes only part of one standard and only the 2D dimension without reference sets. PostGIS implements two standards almost entirely, but there are some difficulties associated with installation and integration. These considerations are illustrated with examples in SQL. Tests were carried out for systems PostgreSQL 8.4.2 + PostGIS 1.5.0 and MySQL 5.1.44. Results for the latest available version of MySQL (Oracle) and for the most recent version published by the previous owners of the project (MySQL AB and Sun) are identical.

REFERENCES

1. IBM DB2 - Spatial Extender l [@:] http://www-01.ibm.com/software/data/spatial/
2. ISO/IEC 13249-3:1999, Information technology – Database languages – SQL Multimedia and Application Packages – Part 3: Spatial, International Organization For Standardization, 2000.
3. Jankiewicz K., Wojciechowski M., 2004. Standard SQL/MM: SQL Multimedia and Application Packages [SQL/MM Standard: SQL Multimedia and Application Packages]. Proceedings of IX Seminarium PLOUG 'Przetwarzanie zaawansowanych struktur danych: Oracle interMedia, Spatial, Text i XML DB', Warszawa, 2004 [in Polish].
4. Melton J., Eisenberg A., 2001. SQL Multimedia and Application Packages (SQL/MM). SIGMOD Record 30(4), 2001.
5. MS SQL 2008. Spatial Storage [@:] http://technet.microsoft.com/en-us/library/bb933790.aspx
6. MySQL Spatial [@:] http://dev.mysql.com/doc/refman/5.1/en/spatial-extensions.html
7. Netzel P., 2009. PostGIS, PostgreSQL. Proceedings of Conference ‘Wolne Oprogramowanie w Geoinformatyce’, Wrocław, 2009 [in Polish].
8. OGC - The Open Geospatial Consortium, [@:] http://www.opengeospatial.org/
9. OpenGIS Implementation Specification for Geographic information – Simple feature access – SQL option. [@:] http://www.opengeospatial.org/standards/sfs
10. Oracle Spatial, [@:] http://www.oracle.com/technology/documentation/spatial.html

11. PostGIS Home Page [@:] http://postgis.refractions.net/
12. PostgreSQL Home Page [@:] http://www.postgresql.org/
13. SQLite Home Page [@:] http://www.sqlite.org/
14. Wojciechowski M., Matuszczak L., 2003. Oracle interMedia na tle standardu SQL/MM i prototypowych systemów multimedialnych baz danych [Oracle interMedia on the background of the standard SQL/MM and prototype multimedia database systems]. Proceedings of IX Conf. PLOUG, Zakopane, 2003 [in Polish].

Adam Piórkowski
Department of Geoinfomatics and Applied Computer Science,
AGH University of Science and Technology, Cracow, Poland
email: pioro@agh.edu.pl

Accepted for print: 13.12.2010