The Journal of Systems and Software 84 (2011) 2314–2323

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Design of a Java spatial extension for relational databases

J. Martinez-Llario*, M. Gonzalez-Alcaide

Department of Cartographic Engineering, Geodesy and Photogrammetry, Universidad Politecnica de Valencia, Camino de Vera s/n 46022, Spain

ARTICLE INFO

Article history:
Received 18 December 2010
Received in revised form 22 June 2011
Accepted 27 June 2011
Available online 6 July 2011

Keywords:
Spatial database
PostGIS
FOSS
Java
GIS
OGC

ABSTRACT

Jaspa (Java Spatial) is a novel spatial extension for relational database management systems (RDBMSs). It is the result of a research project that aims to accomplish two goals: firstly, to fill the absence in the Free and Open Source Software (FOSS) world of a solid Java-based alternative to PostGIS, the leading spatial extension written in C. Secondly, to exploit the advantages of Java and the Java geospatial libraries over C in terms of portability and easiness to extend. Java programming for geospatial purposes is a flowering field and similar solutions to Jaspa have recently appeared, but none of them can equate with PostGIS due to lack of functionalities. Jaspa currently implements almost all PostGIS functionality. The next step will be the enrichment of the software with more sophisticated features: storage of spatial data in a topological structure within the RDBMS, cluster tolerance and geodetic functionalities. Jaspa is being developed at the Department of Cartographic Engineering, Geodesy and Photogrammetry of the Universidad Politécnica de Valencia and it has been published under an Open Source license on the OSOR.eu repository. This paper has been written by its creators with the aim of introducing users to its main capabilities.

1. Introduction

Jaspa is a FOSS software fully written in Java, consisting in a spatial extension for RDBMSs. Such a brief description implicitly covers issues associated to programming languages, GIS architectures or the usage of standards in the geospatial software industry. These topics constitute the framework where Jaspa is born and are subsequently unfolded to understand the need of this new product.

1.1. Usage of standards

Nowadays there is a wide and constantly increasing range of software projects related to the geographical information management. When new software is designed from scratch, it is necessary to implement the standards that define the basic features that will make possible the integration of this new product with the existing ones and with those that will come in the future.

Standards provide three primary benefits for spatial data users: portability, maintainability and interoperability (Groot and McLaughlin, 2000). Using them is a warranty of efficiency, as they save time by removing the need for reinventing approaches to store and handle spatial data (Kralidis, 2008).

The international reference for spatial data management standards is the Open Geospatial Consortium, which released in 1999 the SQL/SFS specification (OpenGIS Consortium, 1999) and its present version released in 2006 (OpenGIS Consortium, 2006). ISO 13249-3, (also referred as "SQL/MM Part 3: Spatial") represents the evolution of the OGC SFS specification, and is the international standard that defines how to store, retrieve and process spatial data using SQL. It defines how spatial data is represented as values, user-defined types, routines and schemas for generic spatial data handling. It also states which functions are available to convert, compare, and process this data in various ways (Stolze, 2003).

Another relevant aspect connected to the development of new software is the software reuse (Fichman and Kemerer, 2001). The usage of existing software equally leads to minimise efforts and thus concentrate the research only in totally innovative features. In this sense Jaspa is largely based in libraries as JTS and GeoTools, both products of proven reliability that support Open standards to ensure interoperability between components.

1.2. System architecture approaches

RDBMSs are used to efficiently maintain and manage large collections of data. Common features to achieve it are transactions, views, referential integrity, indexing and so on. However, regular backend RDBMSs do not support spatial capabilities,

* Corresponding author. Tel.: +34 963877007x75599.
E-mail address: jomarlla@cgf.upv.es (J. Martinez-Llario).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.06.072

Table 1
State of the spatial extensions.

Lang.	Spatial extension	Backend RDBMS	Initial release	Architecture	Functionality	Standards	Spatial index
C	PostGIS	PostgreSQL	2001	Integrated	Full	OGC SFS SQL-MM	GIST
	Ingress geospatial	Ingress	2010	Integrated	Limited	OGC SFS	Yes
	MySQL spatial	MySQL	2004	Integrated	Limited	SQL with geometry types. Not precise spatial operations	MyISAM tables R-tree indexes
	SpatiaLite	SQLite		Layered	Full	OGC-SFS via GEOS	Rtree via SQLite
Java	Spatial DBBox	–	2005		JTS	–	No
	H2 Spatial	H2	2008	Integrated	JTS	–	No
	Hatbox	H2 Derby	2009	Layered	Limited	Partial OGC SFS	Rtree
	GeoDB	H2	2010	Integrated	Limited	Partial OGC SFS	Rtree via HatBox
	Jaspa	PostgreSQL H2	2010	Integrated	Full	OGC SFS SQL-MM	GIST No

especially open source projects. To solve this problem, (Vijlbrief and van Oosterom, 1992) states three kinds of system architectures that have been used to link the spatial data to the RDBMS.

The first approach is the dual architecture, which stores spatial data outside the RDBMS (e.g., Esri ARC/INFO or Intergraph MGE). The second one, the layered architecture, translates the spatial data into the relational model (e.g., GeoView or SIRO-DBMS). The third approach is the integrated architecture, which consists on extending the RDBMS with spatial data types and functions to manage geographic features (e.g., TIGRIS or PostGIS).

In the dual architecture the databases are used as a mere container of information, and the spatial operations are performed by frontend applications. Consequently, this architecture does not take advantage of the well-known RDBMS ACID properties. This inconvenient is solved in the layered architecture, but the retrieval and accessing methods are more complex and the system is overall less efficient.

In the integrated architecture the required geometric data types and functions are implemented in the RDBMS kernel. This approach has been proven to be the most cost-effective, but to accomplish it an extensible RDBMS is necessary (Vijlbrief and van Oosterom, 1992). As a result, several spatial extensions for RDBMS have been developed, which are described in Section 2.

1.3. Possibilities of Jaspa

The main goal of this paper is to show to the GIS community the architecture, design and capabilities of a new spatial database which tries to provide new features and some advantages compared with the current spatial database solutions.

The authors want to demonstrate in this paper that Jaspa is easy to extend with new spatial functionalities which bring many possibilities to researchers as it is shown in Section 7.

This paper does not aim to focus in some special features as the topology rules system or the geodesic functions but to show Jaspa as a whole product and to prove to the readers that it could be an appropriate platform for researching and developing new cartographic algorithms, cartographic models, etc.

2. State of spatial extensions

GIS software is a very dynamic world and new database projects arise quite often. In order to compare Jaspa capabilities with already released products, Table 1 (Ramsey, 2007) summarizes common FOSS spatial extension topics, which are then explained.

As it is shown in Table 1, C projects are considerably more mature. Nevertheless, several reasons encourage the need of a spatial extension fully written in Java. Jaspa aims to be a product that can be easily extended by users, creating their own functionalities adjusted to their own needs. Furthermore, it can be more effectively integrated with other Java products, which are taking now the GIS sector and thus dispense with the use of the Java Database Connectivity. Finally, it can directly access to the functionalities of JTS and many other Java geospatial solutions as GeoTools, gvSIG, Kosmo, uDig or GeoServer. On the other hand, the Java world lacks an Open Source spatial database that can be further developed by expert Java users. Jaspa is born in this context, and its main goal is to fill that gap.

The state of the art shows that Oracle Spatial and PostGIS are the most consolidated spatial extensions for RDBMSs (Martinez-Llario et al., 2009). We will focus subsequently on the latter as it has been the spatial extension that leads the Open source world and is the main reference for Jaspa.

2.1. C approaches

PostGIS provides spatial capabilities to PostgreSQL, including geometric data types, wide range of functionalities and spatial indexing among many others. It is by far the most widespread spatial extension, therefore many GIS applications such GIS desktops or map servers are capable of working with data stored in PostgreSQL by using PostGIS (Ramsey, 2007).

2.2. Java approaches

The GeoServer team has primarily carried out experimental developments in this field. In 2005 appeared SpatialDBBox[1] as a method for adding spatial functions to non-spatial databases. This project defined the methodology to use the library JTS for the implementation of all geometry functions in different databases.

Later GeoServer released H2 Spatial[2], a spatial extension based on the application of the SpatialDBBox ideas. H2 Spatial was the first to use H2 as a backend RDBMS relying on an integrated architecture. Its main drawbacks were limited functionality and lack of spatial indexing. Despite these weaknesses, the release of H2 Spatial was

[1] http://geoserver.org/display/GEOS/SpatialDBBox.
[2] http://geoserver.org/display/GEOS/H2+Spatial+Database.