JPOX-Spatial - Project Summary

Thomas Marti
Stefan Schmid

University of Applied Sciences, Rapperswil (HSR)
June 28th, 2006

Documentation, source code and latest releases are available on the project website:
http://jpoxspatial.zones.ch/

Introduction

An increasing number of today's RDBMSs (relational database management systems) have integrated support for spatial data types like point, line or polygon. This enables developers to write geospatial applications, while taking advantage of most RDBMS' core capabilities like transaction, security or persistence. The standard way to access a database for a Java developer is to use the JDBC driver, which is usually provided by the RDBMS vendor. The main disadvantage of this approach is that developers have to worry about O/R mapping themselves. General-purpose persistence frameworks like Hibernate or JDX and also spatial-specific frameworks like GeoTools or OpenMap attempt to fill this void. By using such a proprietary or non-standard compliant persistence layer, a developer is restricted to use the chosen API and is unable to move effortlessly to an alternative implementation.

Sun's JDO specification provides a technology for transparently persisting Java objects (POJOs). This standardized, object-oriented persistence API promises high usability and performance. Unfortunately the JDO spec. only demands mandatory support for basic Java types like `int`, `String` or `Date`. Because of this, support for spatial data types in current JDO implementations is sparse. However, most implementations provide plug-in mechanisms to add support for user defined data types.

JPOX is an open source and fully compliant JDO implementation that was chosen by SUN as reference implementation for the JDO 2.0 specification. It supports all of the major RDBMSs[1] on the market today and allows users to define their own types. Users may also extend the query language with user defined methods.

In addition, JPOX is maintained by two very enthusiastic core developers and has an uncluttered, well-refactored code base. Consequently, we chose JPOX for our term project.

Vision

Our vision is to provide industry standard persistence for any geospatial application in an environment that supports the OGC[2] 'Simple Features For SQL' specification[3]. We want to provide a plug-and-play approach with the application's geometry object model as well as with the underlying RDBMS. This means users should be able to easily replace the RDBMS and/or the geometry library as long as they are OGC SFS compliant.

[1] Oracle, IBM DB2, MySQL, PostgreSQL, MS SQL, etc. See http://www.jpox.org/docs/1_1/rdbms.html for the complete list.
[2] Open Geospatial Consortium
[3] See http://www.opengis.org/docs/99-049.pdf

Goals

The goal of JPOX-Spatial is to allow the use of JPOX as a persistence layer for geospatial applications. To achieve this we have to extend several parts of JPOX:

- Define type mappings to let JPOX know how to persist spatial Java types.
- Extend the query language with spatial functions according to the OGC SFS specification.
- Adapt JPOX for the specifics of spatial databases.

The vast majority of current open source GIS projects in the Java world use the JTS Topology Suite[4] as basis for geometry representation. On the backend side, PostGIS has become the standard spatial database for open source GIS tools. PostGIS adds spatial database capabilities to PostgreSQL. On this account, support for JTS and PostGIS is a must-have to meet the requirements of the open source community. However, in commercial environments use of Oracle, IBM DB2 and other RDBMSs is widespread. It is therefore desirable to add support for further database systems.

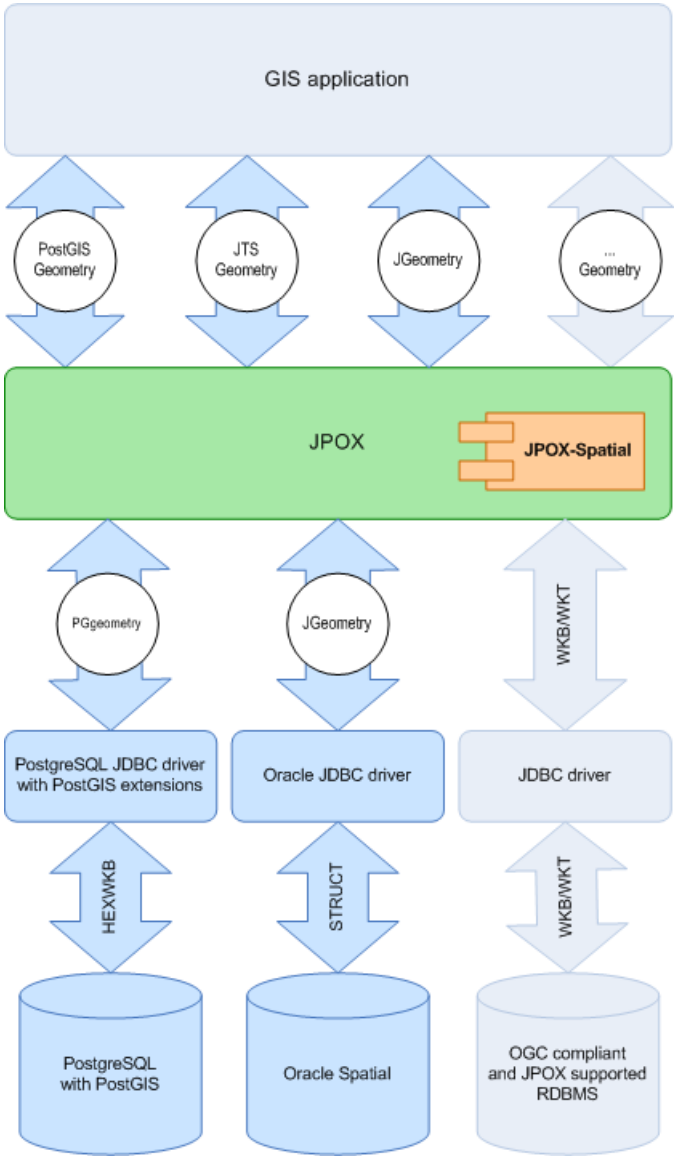

Figure 1. Spatially enabled JPOX forms a persistence layer for geospatial applications

[4] http://jump-project.org/project.php?PID=JTS&SID=OVER

Results

The following table shows the current status of the project:

	PostGIS	MySQL	Oracle	DB2
PostGIS Geometries	✓	✓	⚠	⚠
JTS Geometries	✓	✓	⚠	⚠
Oracle JGeometry	⚠	⚠	⚠ *	⚠

Table 1. Geometry object model to database mappings

*** Will be supported after merging with the existing JPOX spatial plug-in.**
✓ Current Release
⚠ Future Releases

In addition to these mappings we also extended the JDO query language (JDOQL) with functions to query spatial data. These functions follow the definitions in OGC SFS and are translated into appropriate SQL statements provided the underlying database system implements the functions and the geometry object model accordingly.

This set of more than forty functions contains:

- basic methods on geometry objects like `IsSimple()` and `Boundary()`
- methods for testing spatial relations between geometric objects like `Intersects()` and `Touches()`
- methods that support spatial analysis like `Union()` and `Difference()`
- methods on geometry types like `X()` on type `Point` and `PointN()` on type `LineString`

For the complete list of all supported functions please see CHANGELOG.txt of the current release.

Related work and further development

During the work on this project, the JPOX team released a plug-in that supports the storage and query of spatial types. At the moment this plug-in supports only Oracle's `JGeometry` and a few functions for comparisons between spatial types. The maintainers agreed to our proposal to merge our efforts. Hence, when this term project is finished we will donate all the sources to JPOX and the plug-in will be prominently featured on the JPOX website.

We are confident, that we can implement all the planned features before the term is finished. But there is still a lot of room for improvement in JPOX-Spatial: A major task will be to implement support for additional RDBMSs (that conform to OGC SFS). Another one to analyse other open source GIS projects like GeoTools or OpenMap and evaluate whether a JPOX mapping for those geometry models is possible and useful. Another line of work is the improvement of the already supported geometry models. Notably the geometry classes from the `org.postgis`-package have great potential for improvement. We are already in contact with one of the maintainers (Markus Schaber) of this package and have discussed different possibilities to enhance the performance and minimize the memory footprint of these classes.

Another possible task, although not directly part of JPOX-Spatial, would be trying to implement a geometry library according to the GeoAPI[5].

[5]GeoAPI is a set of Java interfaces based on the ISO 19107 specification. See http://geoapi.sourceforge.net/