HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

INSTITUT
FÜR
SOFTWARE

JPOX-Spatial - Persistence Framework For Spatial Applications

S.F. Keller, A. Kälin, Th. Marti and S. Schmid

GISpunkt / Institute for Software

University of Applied Sciences Rapperswil (HSR)

CH-8640 Rapperswil, Switzerland

www.gis.hsr.ch / www.ifs.hsr.ch

Use Cases

Developers want ...

- ... to use a comprehensive set of spatial data types in their Java applications.
- ... a persistence solution for spatial objects without technology gap.
- ... to use complex spatial functions (according to the OGC Simple Features specification).

JDO and JPOX

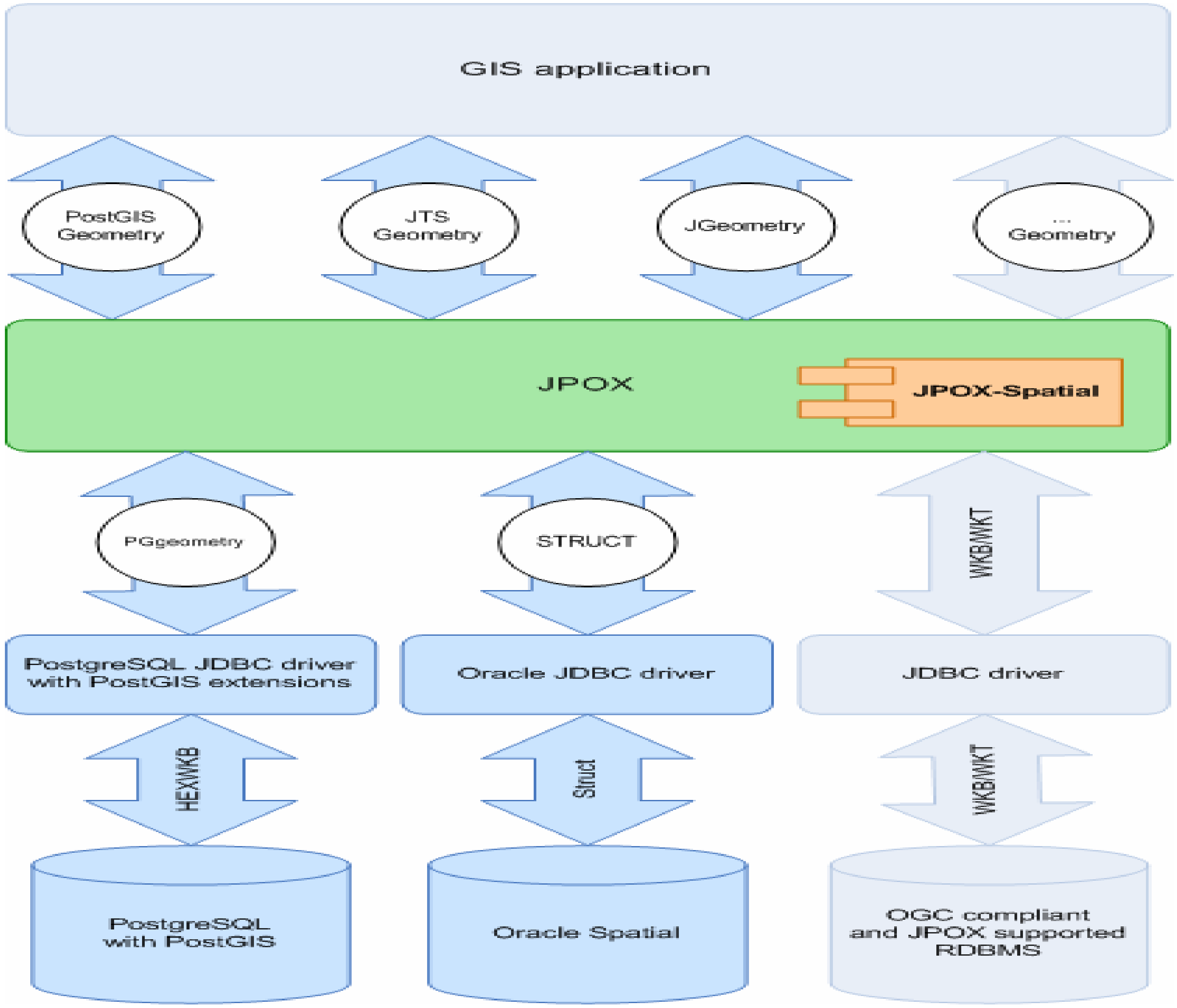
- Java Data Objects (JDO)
  - Transparent persistence of Java objects
  - Persistence to all major RDBMS and all main ORM patterns, e.g. allows querying using either JDOQL or SQL
- JPOX
  - Free, compliant implementation of the JDO specs, comes with own byte-code enhancer
  - Outperforms some other O/R mappers
  - Will implement JPA spec (part of EJB3)
  - JPOX 1.1.0-final is JDO 2 Reference implement.

JPOX and JPOX-Spatial

- Idea: Get geometry types in as user defined types
- Authors: Thomas Marti and Stefan Schmid
- Semester thesis project, summer 2006, Master study in Informatics, University of Applied Sciences Rapperswil UAS-HSR, Switzerland
- 14 weeks, ~650 hours

http://www.hsr.ch

JPOX-Spatial Overview (1/3)



JPOX-Spatial Overview (2/3)

Databases supported

DB / Geometry Lib.	PostGIS	MySQL	Oracle	DB2
PostGIS Geometry Lib.	Ok	Ok	*	*
JTS Geometry Lib.	Ok	Ok	*	*
Oracle JGeometry Lib.	*	*	soon	*

JPOX-Spatial Overview (3/3)

- **Geometry classes:**
  - **PostGIS Geometry**
  - **JTS Geometry**
  - **Oracle JGeometry, more...?**
- **Mapping:**
  - **Forward, Reverse & Meet-in-the-middle**
- **Approach:**
  - **Byte code enhancement to support `PersistenceCapable` interface of classes**
  - **Loading-on-access**

Preparation of JPOX-Spatial

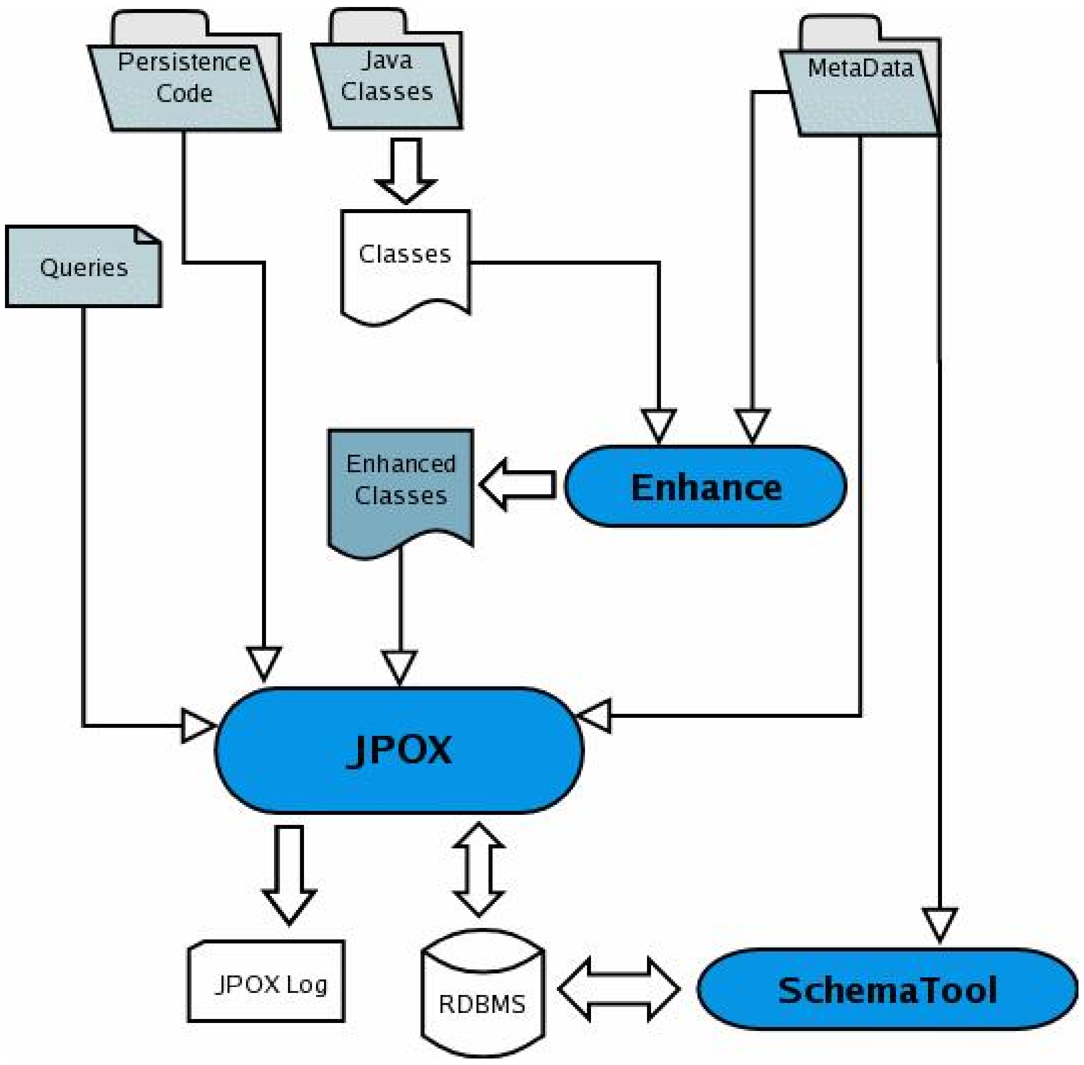
- **Choose the right JAR file for your mapping scenario:**
  - `jts2mysql` ➔ `jpoxspatial-jts2mysql-<version>.jar`
  - `jts2postgis` ➔ `jpoxspatial-jts2postgis-<version>.jar`
  - `pg2mysql` ➔ `jpoxspatial-pg2mysql-<version>.jar`
  - `pg2postgis` ➔ `jpoxspatial-pg2postgis-<version>.jar`

- **Additional JAR files are needed for every mapping scenario. Examples:**
  - `jts2mysql`
    - `jts.jar`
    - `mysql-connector-java-<version>.jar`
  - `pg2postgis`
    - `postgresql-<version>.jdbc3.jar`
    - `postgis.jar`

Installation & Use of JPOX-Spatial

- Make all needed JAR files available in the classpath.
- Use the Java geometry types of your chosen library (e.g. JTS) in your application.
- Specify the geometry fields in your meta-data as you would with any other JPOX-supported data type.
- Enhance your classes using JPOX’ Enhancer Tool.
- Persist and query your spatial data...

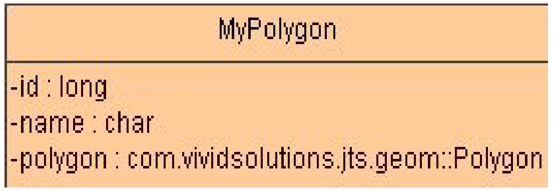
JPOX Process



Source: http://www.jpox.org

Example: JTS - PostGIS

Metadata



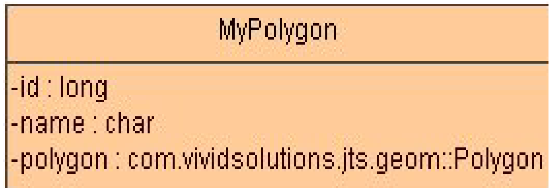
MyPolygon

-id : long
-name : char
-polygon : com.vividsolutions.jts.geom::Polygon

```
<jdo>
    <package name="ch.hsr.foss4g">
        <class name="MyPolygon" detachable="true">
            <extension vendor-name="jpox" key="postgis-srid" value="-1"/>
            <extension vendor-name="jpox" key="postgis-dimension" value="2"/>
            <field name="id"/>
            <field name="name"/>
            <field name="polygon" persistence-modifier="persistent"/>
        </class>
    </package>
</jdo>
```

- **postgis-srid and postgis-dimension parameters are only used when PostGIS is the backend. The given values will be used to create spatial columns with the `AddGeometryColumn()` function.**

Example: JTS - PostGIS



Create and persist an object

```
PersistenceManager pm;
Transaction tx = pm.currentTransaction();

Polygon polygon = (Polygon)wktReader.read(
    "POLYGON((0 0,3 0,3 3,0 3,0 0),(1 1,5 1,5 5,1 5,1 1))" );
MyPolygon myPolygon = new MyPolygon( 1, ‘a‘, polygon );

tx.begin();
pm.makePersistent( myPolygon );
Object id = pm.getObjectId( myPolygon );
tx.commit();
```

Retrieve object from datastore

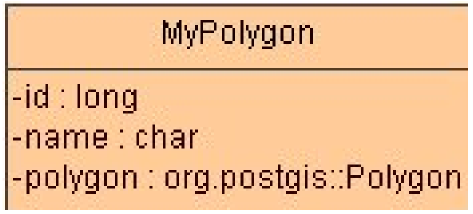
```
MyPolygon myPolyFromDatastore = (MyPolygon)pm.getObjectById( id );
```

Queries

- **JPOX-Spatial extends 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.**
- **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

JDOQL Example: PostGIS - PostgreSQL

- „Return all MyPolygons where point (30, 30) is spatially within the polygon“
- Backend is PostgreSQL/PostGIS
- Application uses geometry objects from PostGIS (JDBC)

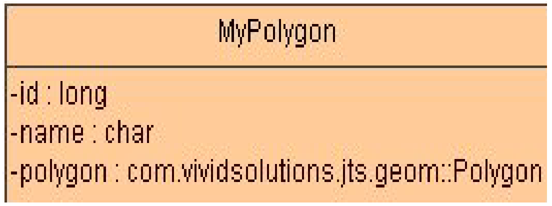


```
Point point = new Point( "SRID=1234;POINT(30 30)" );
Query query = pm.newQuery( MyPolygon.class,
      "OGCSF.within( :point, polygon )" );

List<MyPolygon> list = (List<MyPolygon>)query.execute( point );
for ( MyPolygon polygon : list ) {
      System.out.println( point + " is within "
                              + polygon.getName() );
}
```

JDOQL Example: JTS - MySQL

- „Return all MyPolygons where point (30, 30) is spatially within the polygon“
- Backend is MySQL
- Application uses geometry objects from JTS



MyPolygon
-id : long -name : char -polygon : com.vividsolutions.jts.geom::Polygon

```
Point point =(Point)wktReader.read( "POINT(30 30)" );
Query query = pm.newQuery( MyPolygon.class,
      "OGCSF.within( :point, polygon )" );

List<MyPolygon> list = (List<MyPolygon>)query.execute( point );
for ( MyPolygon polygon : list ) {
      System.out.println( point + " is within "
                         + polygon.getName() );
}
```

Further development of JPOX

- „In the next versions we will have Multiple API, Datastore Agnostic, RDBMS Agnostic, Query Languages, Multiple Types, Pluggable and Managebility aspects implemented.“ […]

- „For longer term, JPOX should be seen as a data access platform providing ORM, Web Services, Multidimensional, Mining, Functional/Technical metadata views of Data.“

-- Erik Bengtson, Core Developer of JPOX

Further development of JPOX-Spatial

- Complete migration and integration into JPOX project
- Support additional datastores (IBM DB2, Oracle,...)
- Support additional geometry libraries
- Implement support for mutable types

JPOX-Spatial: Infos and Download

- JPOX
  - Home: www.jpox.org
  - Download: www.jpox.org/docs/download.html
  - Open Source Apache 2 license, currently JPOX 1.1.2

- Next release of JPOX-Spatial:
  - Cooperation with Eisenhut Informatik (Suisse) and Refractions Research (Canada)
  - JPOX-Spatial will be an official part of JPOX that can be downloaded from the plugin site (see download)