OSGeo Journal

The Journal of the Open Source Geospatial Foundation Volume 3 / December 2007

Proceedings of FOSS4G 2007

to support smooth panning.

Bibliography

[1] P. H. Dana Map Projection Overview http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj_f.html

[2] DM Solutions Group Inc. PHP MapScript http://www.maptools.org

[3] R. Kingston (1998) Web-based GIS for public participation decision making. In *Procs* of NCGIA PPGIS Meeting, Santa Barbara, California. Retrieved Map 2003 from http://www.ncgia.ucsb.edu/varenius/ppgis/papers/kingston/kingston.html

[4] Eum D. and Minoura T. (June 2003) Web-based database application generater. *IEICE Transactions on Information and Systems, Vol. E86-D, No. 6.*

[5] Fogelsong, C. (2002) Biotics 4.0 data model version 1.0. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/HDMS-DataModel.shtml

[6] McKenna, Jeff MapServer PHP/MapScript Class Reference - Versions 3.6, 4.0 & 4.2 DM Solutions Group Inc.

[7] NatureServe (February 2002) Element Occurrence Data Standard. Retrieved January 4, 2004, from http://whiteoak.natureserve.org/eodraft/all.pdf

[8] NatureServe (December 2003) Biotics 4.0 Getting Started Guide. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/biotics-learn-more.shtml (now obsolete).

[9] Sano J., Wanalertlak N., Maki A., and Minoura T. (July 2003) Benefits of web-based GIS/database applications. In *Prosc. of 2nd Annual Public Participation GIS Conference* Portland, Oregon.

[10] USDA-ARS Western Regional Plant Introduction Station, USDA - Agricultural Research Service, Pullman, Washington http://www.ars-grin.gov/ars/PacWest/Pullman/

[11] Ramsey, Paul PostGIS Manual Refractions Research Inc.

[12] University of Minnesota (2003) MapServer http://mapserver.gis.umn.edu

[13] Sharma, A. (December 2003) Web-based analysis module for a germplasm collection. Master of Science report, School of Electrical Engineering and Computer Science, Oregon State University

[14] Wangmutitakul P, Li L, and Minoura T. (March 2003) User Participatory Web-Based GIS/Database Application. *Proc. of Geotec Event Conference*

[15] Wangmutitakul Paphun et al. (2004) Framework for Web-based GIS/database Applications *Journal of Object Technology 3*, 209-225

[16] Wuttiwat T., Minoura T. and Steiner J. (May 2003) Using Digital Orthographic Aerial Images as User Interfaces *Proc. of ASPRS Annual Conference*, Anchorage, Alaska

Toshimi Minoura, Nirut Chalainanont, Junya Sano
Oregon State University
http://engr.oregonstate.edu/~minoura/research/creeda/
minoura AT eecs.orst.edu

db4o2D - Object Database Extension for 2D Geospatial Types

Stefan Keller

db4o stands for "database for objects". It's a native object oriented database management system (OODBMS) written in Java and .NET and thus targeted towards these two platforms. The software was first released 2001 by db4objects, Inc., and since then it got a major market share among the so called second generation object databases. It is available under two licenses ("dual licensing model"), an open source licence of type GPL for personal and non-commercial use as well as a commercial license.

In this contribution we first introduce db4o. In the chapter *OODBMS and db4o* we discuss the advantages, limitations and differences from a conceptual and a programmer's perspective. Then we report

about a student thesis project called db4o2D. Finally there is a conclusion with an outlook of the project db4o2D.

First steps in db4o

Let's start with some source code in order to give an idea how easy it is to persist application objects. Our first example demonstrates the four well known steps from database technology: Create, Read, Update and Delete (CRUD). The example is about Person objects which consist of last name, first name and year-of-birth:

```
// 1: Initialize an object container
ObjectContainer db= null;
try {
    // 2: Open db4o database (embedded mode)
    db= Db4o.openFile("addressbook.yap");

    // 3: Create and store some 'Kellers'
    db.set(new Person("Brian", "Keller", 1960));
    db.set(new Person("Clara", "Keller"));
    db.set(new Person("Test"));

    db.commit();

    // 4: Find and read all Kellers
    // with Query by Example
    ObjectSet result= db.get( \
        new Person(null, "Keller", 0));
    while (result.hasNext())
        System.out.println( \
            (Person) result.next());

    // 5: Update Clara's age
    result= db.get(new Person("Clara", "Keller"));
    Person found= (Person) result.next();
    found.setYearOfBirth(1970);
    db.set(found);

    // 6: Delete Test data
    result= db.get(new Person("Test"));
    while (result.hasNext())
        db.delete(result.next());

    // 7: Commit all
    db.commit();
} finally {
    if (db != null)
        db.close();
}
```

Fig. 1. Code fragment showing CRUD operations on person objects stored in an address book.

In this example we go though the following steps which show some CRUD operations:

1. The instruction following this comment initializes a container where db4o manages the set of objects to be persisted in a transactional way.
2. The second instruction opens a db4o database file. We propose to use the file extension .yap but this is not normalized. There is an official explanation which says that this is the abbreviation of "yet another protocol". Unofficially it's referring to Yap, a tiny island of Micronesia. As indicated we choose to use db4o in embedded mode. There exists also a client/server mode.
3. With the `set()` method of an `ObjectContainer` three newly created Person objects are stored. With `commit()` all created, changed or deleted objects are forced to be synchronized with the database file.
4. In this step we fetch all objects from the database which are equal to the last name "Keller" and iterate over the result. In this example "Query by Example" is used from the three query languages available from db4o.
5. When creating "Clara Keller" in step 3 we omitted the year-of-birth in the Person's constructor (since it's polite not to reveal the age of a woman beforehand). Now we insist in setting this value, so we have to look for objects like "Clara Keller" and update the first one the query gives back. We use again the set() method assuming that there exists a `setYearOfBirth()` method in the Person class definition.
6. For demonstration purposes we previously inserted also some test data which will be deleted in this code section. This time all objects retrieved by the query will be cleaned up with the `delete()` method of the `ObjectContainer`.
7. Finally we conclude with the `commit()` method and close the database.

The only definitions which are missing in this code fragment are the import statements as well as the Person class consisting of three constructors, the `setYearOfBirth()` method and optionally an overridden `toString()` method in order to print out Person objects nicely.

That's all to demonstrate the simplicity of a db4o enabled application. Next we explain some use cases of OODBMS, some additional features of db4o before an obvious extension is presented which will add geometry types to db4o.

OODBMS and db4o

Use Cases and Features

When there is only one application running on top of a database at a time and if it's a mobile application like in location based systems (LBS) or embedded software it's perhaps worthwhile to evaluate one of the so called second generation OODBMS. Because of the easy management of object relationships OODBMS are intrinsically well suited when complex object models, flat object structures or tall object trees are involved – which actually is often the case in Geographic Information Systems (GIS) and LBS.

These are some technical features of db4o:

- Embedded mode and client/server mode.
- No runtime server administration. Database properties are controlled out of the host application.
- Small space requirements of the program library on disk and in memory at runtime.
- Ease of use: db4o is using reflection application programming interfaces (API) from Java and .NET. So there are no extra annotations, no pre- or post-processing (byte code engineering), no sub classing nor interface implementation.
- Methods to control lazy loading of nested object relationships (depth).
- Replication tools as add-on.

Like one would expect from a database, db4o implements ACID (atomicity, consistency, isolation and durability) properties which guarantee reliable transactions: A transaction starts when opening or querying the database and ends with `commit()` and `rollback()` methods. Three approaches for queries are implemented: Query by Example, Simple Object Database Access (SODA Criteria) queries and "Native Queries". The first one was shown above. SODA queries are much like those used in object-relational (O/R) mapping frameworks. Some of the theory behind Native Queries stems from a project from Microsoft (LINQ 2007).

Advantages

Speaking of O/R mapping frameworks we have to compare OODBMS with this technique too. db4o is very fast according to benchmarks (Polepos 2007). A conceptual argument is that there is no object / relational impedance mismatch: No data types mapping and wrapping (unless wanted), no creation of separate relational schemas, no SQL dialects and no plain SQL query strings.

So it's important to mention that db4o offers decent support of agile software development techniques and refactoring: This is because of queries are written in the host language (Java, .NET) and thus are type safe. There are also nice schema evolution features and no SQL. Software engineers have an easier life than before because they reside in the object world as opposite to database professionals' world.

Limitations

db4o is probably not well-suited for being used in large data warehouses and in data mining. It's typically not recommended when several application are accessing the database with many views. One can see from the different query languages that there is no single standard and mature query language available compared to the pre-dominant SQL from the relational paradigm. Constraints like referential integrity are not (yet) part of any language except inside a Native Query which basically implements a call back function. Finally, the current lack of standardization was recognized. There are activities from the Object Management Group (OMG) in order to work towards a new release of the ODMG standard version 4.

Project db4o2D

PlaceLab (Intel 2006) — an Intel project — delivered mobile and desktop applications about a wireless LAN (WiFi) positioning. The database behind this software is a relational open source embedded database. A crucial component there is the management of access points together with their positions. This serves us here as a showcase of an object database which replaces the existing relational database and stores geometries as first class objects.

In a thesis project (db4o2D 2007) there was decided to use db4o and to adapt the broadly used Java Topology Suite library (JTS 2007). JTS follows the "Simple Features" standard (OGC 2006) which defines four 2D (2.5D) geometry attribute types: Point, LineString and Polygon as well as stable operations on it.

So in the following code fragment (c.f. Fig. 2) it is shown how wireless access points are created (2), stored (3) and reread (4). A point contains a coordinate value pair and includes a default coordinate reference system and measurement units.

```
// 1: Open db4o database (embedded mode)
db= Db4o.openFile("poidb.yap");
```

```
// 2: Prepare two JTS points
GeometryFactory factory= new GeometryFactory();
double latitude1= 47.225571, longitude1= 8.822271;
Point pt1= factory.createPoint(new Coordinate(
        longitude1, latitude1));
double latitude2= 47.225582, longitude2= 8.822282;
Point pt2= factory.createPoint(new Coordinate(
        longitude2, latitude2));

// 3: Create and store the access points
db.set(new AccessPoint( \
    1001, "802.11g", "wep", 7, pt1));
db.set(new AccessPoint( \
    1002, "802.11b", "open", 11, pt2));
db.commit();

// 4: Iterate over all access points
ObjectSet result= db.get(new AccessPoint());
while (result.hasNext()) {
    System.out.println( \
    (AccessPoint) result.next());
}
```

Fig. 2. Code fragment with wireless access points which contain point geometry.

Conclusion

The db4o2D project will become an add-on to db4o and is still ongoing. Although that the existing database access layer already was well separated it becomes obvious how smaller, simple and better to maintain the code becomes by using a pure OODBMS.

Future work includes a geospatial index and stress tests for large databases as well as for multi-threading client/server mode. Complex SODA queries is another open issue. It's noteworthy to state that this is no obstacle so far because Native Queries can be used in the meantime.

In order to disseminate object database technologies a non-profit group ODBMS.ORG (2007) was created. db4o is one of the leading object database projects and it tries to solve well known problems in embedded software, software engineering, LBS and GIS. Given that these problems have been around for quite some time one could say that second generation OODBMS like this are something like going back to the future.

References

db4o (2007): Homepage of db4o and db4objects, Inc.: www.db4o.com (last visited October 22nd, 2007).

db4o2D (2007): db4o2D project space, http://developer.db4o.com (last visited October 22nd, 2007).

Intel (2006): The PlaceLab Project, PlaceLab — A privacy observant location system, www.placelab.org (last visited October 22nd, 2007).

JavaWPS (2007): A positioning system in Java based on Wireless Local Area Network signals, www.gis.hsr.ch/wiki/JavaWPS (last visited October 22nd, 2007).

JTS (2007): Java Topology Suite, an API of 2D spatial predicates and functions, www.vividsolutions.com/jts/JTSHome.htm (last visited October 22nd, 2007).

LINQ (2007): The LINQ project, http://msdn.microsoft.com/netframework/future/linq (last visited October 22nd, 2007).

ODBMS.ORG (2007): A vendor-independent resource portal on object database technology, www.odbms.org, (last visited October 22nd, 2007). OGC (2006): OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture, www.opengeospatial.org/standards/sfa (last visited October 22nd, 2007)

PolePosition (2007): An open source database benchmark, www.polepos.org (last visited October 22nd, 2007).

Stefan Keller
Institute for Software and GISpunkt
University of Applied Sciences Rapperswil (UAS HSR)
CH-8640 Rapperswil, Switzerland
www.ifs.hsr.ch
Stefan Keller is professor for information systems teaching database technologies and programming.
stefan.keller AT hsr.ch

The Open Source Geospatial Foundation, or OSGeo, is a not-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. It also serves as an independent legal entity to which community members can contribute code, funding and other resources, secure in the knowledge that their contributions will be maintained for public benefit. OSGeo also serves as an outreach and advocacy organization for the open source geospatial community, and provides a common forum and shared infrastructure for improving cross-project collaboration.

Published by OSGeo, the OSGeo Journal is focused on presenting discussion papers, case studies and introductions and concepts relating to open source and geospatial software topics.

Acknowledgements
Various reviewers & the GRASS News Project

The *OSGeo Journal* is a publication of the *OSGeo Foundation*. The base of this journal, the LaTeX 2_ε style source has been kindly provided by the GRASS and R News editorial board.

Journal online: http://www.osgeo.org/journal
OSGeo Homepage: http://www.osgeo.org
Postal mail: OSGeo
PO Box 4844, Williams Lake,
British Columbia, Canada, V2G 2V8
Telephone: +1-250-277-1621

ISSN 1994-1897