ODBMS.ORG User Report No. 16/08
Editor Roberto V. Zicari- ODBMS.ORG *www.odbms.org*
August 2008.

Category: **Academia**
Domain: *Research/Education*
User Name: **Stefan Keller**
Title: *Professor*
Organization: **HSR Rapperswil, Switzerland**.

Stefan Keller is a professor in information systems at the computer science department of the University of Applied Sciences Rapperswil since 2001.
Prior to his tenure he lead the realization of an infrastructure for integration of public/private geographic information systems in Switzerland. Before he worked for a fortune 500 technology services and solutions company.

Q1. Please explain briefly what are your application domains and your role in the enterprise.

Stefan Keller: I'm giving introductory courses on programming (Java) and advanced database management systems.
My research and application domain is mainly geographic information systems (GIS).
One says that 80% of government and enterprise data has a geographic component and this potential is just beginning to be exploited leading to a promising business growth. GIS often are treated separate from mainstream IT - for example when implementing data warehouses - as GIS are non-standard database systems extended with computer graphics components. Now that map mash-ups and mobile phones equipped with GPS are widespread, location-aware technology is becoming at least more visible. GIS have quite some peculiarities, like expensive data capturing, complex constraints, large volumes of data and often complex data structures.

Q2. When the data models used to persistently store data (whether file systems or database management systems) and the data models used to write programs against the data (C++, Smalltalk, Visual Basic, Java, C#) are different, this is referred to as the "impedance mismatch" problem. Do you have an "impedance mismatch" problem?

Stefan Keller: Yes, there are plenty of mismatch problems on basic

data types which are even aggravated when there are complex data structures. Take coordinates as a simple example: Coordinates have a mandatory northing and an easting value as well as an optional height component sharing common metadata (e.g. coordinate system).
How do you model this in an relational database management system (RDBMS) unless the product offers a spatial extension (which the major RDBMS products do)?

Mismatching data types - as suggested by the definition in the question - is not the only aspect: Firstly there is the 'cultural impedance mismatch' as Scott Ambler coined it (ODBMS.ORG User Report 9/08: Scott Ambler, http://www.odbms.org/downloads.html#odbms_ur). In public agencies which still constitute a large part of the GIS market the 'cultural impedance mismatch' is hardly to overcome because typically RDBMS products are part of their general conditions.

Secondly, I'm often observing mismatches induced by hierarchically modeled objects. The classification structure of points of interests (POIs) is such an example.

Q3. What solution(s) do you use for storing and managing persistence objects? What experience do you have in using the various options available for persistence for new projects? What are the lessons learned in using such solution(s)?

Stefan Keller: In teaching I'm using mainly Oracle RDBMS, PostgreSQL and db4o as well as JPOX, Hibernate and of course JDBC.
In a recent project about a social networking platform for sharing whereabouts of friends, we had to choose a database.
We have chosen PostgreSQL because it's perhaps the most mature open-source database. The main queries returned lists like nearby friends and POIs, so we implemented it with stored procedures. Discussions arose around the mapping of POI categories as explained above. The lesson learned here was, how difficult it was to convince even open-source programmers to switch the database they know well to another one.
We're still way apart from where industrial hardware engineering is, where one chooses the "right tool for the right task".

It's perhaps worthwhile to note that by using stored procedures we certainly utilized databases for more than persistence solution but also as

part of the domain/business layer. These stored procedures defined a programming interface (API) which allowed us to switch from Java to PHP and back.

In another students project we tried to extend db4o with user defined (complex) geometry types and a spatial index. There we got problems with the basic extension mechanisms as these have been fundamentally refactored in the early versions 6.x of db4o. Finally, we had to postpone the implementation because of time constraints.
This showed us that many object database management systems (ODBMS) strive for stability. We'll try later again still emphasizing that ODBMS often are more suitable for complex data types than RDBMs.

Q4. Do you believe that Object Database systems are a suitable solution to the "object persistence" problem? If yes why? If not, why?

Stefan Keller: I think that they contribute to it and still deserve closer attention when evaluating an object persistence solution.
ODBMS to me fit well, when they are exclusively used as a persistence solution out of an integrated development environment in one programming language and when complex data types are important, as well when there are mostly navigational queries.
But their fit-for-use still degrades when there are 'join' queries needed between unrelated objects, e.g. when there POIs are queried around a position anywhere on earth. Then, I've the impression that many ODBMS can't cope with large data sets (> 1GB). I also think, that they too tightly couple with the business layer of a particular (client) application.

Q5. What would you wish as new research/development in the area of Object Persistence in the next 12-24 months?

Stefan Keller: I would like to see much more research/development in exploring and standardizing query languages, especially in the direction of language integrated query languages, and notably to make it independent from .NET ("LINQ for Java").

I've followed some discussions on ODMG.org (www.odbms.org) and I hope they will agree on a pragmatic, syntactically/computationally simple but still powerful approach. This would include polymorphic operators, which

allow an ad-hoc joining of objects which aren't associated with references before hand.

Let me mention in this context a "LINQ for Java" implementation called JaQu which stands for "Java Query" (see www.h2database.com).

For example this SQL statement "SELECT * FROM PRODUCTS P WHERE P.UNITS_IN_STOCK = 0" shows like this in JaQu:

```
1: Product p = new Product();
2: List<Product> soldOutProducts =
3:   db.from(p).where(p.unitsInStock).is(0).select();
```

This for example doesn't need Java language changes (as they did in .NET and as some are proposing also for Java in ODMG.org) – although closures could probably help. JaQu simply seems to be a pragmatic "proof-of-concept" which could contribute to the query standardization debate.

##