ODBMS.ORG User Report No.5/08
Editor Roberto V. Zicari- ODBMS.ORG *www.odbms.org*

July 2008.

Category: **Academia**
Domain: **Research/Education**
User Name: **Stefan Edlich**
Title: Professor
Organization: **TFH-Berlin, Germany**

Stefan Edlich is a professor at TFH-Berlin since 2005. Prior to his tenure in Berlin he was professor at the University of Brandenburg (App.Sc.) since 2002.
From 1995 to 1998 he was the head of an IT-Department (trade-supervision) of a German top 10 Bank. In 1999 he joined one of the most successful OO Consulting-Companies in Germany. Simultaneously he worked for BEA-Systems Inc. as a professional trainer teaching the BEA product portfolio (Weblogic et al) to German Fortune-100s.

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

[Stefan Edlich] Around the millennium I did consulting with and around the BEA Weblogic portfolio. Customers were mostly big German companies with big enterprise databases and a huge amount of transactions to be processed. Like banks or companies in the energy sector using often Oracle as the background database. After switching my career to science and education, this has slightly changed. I have lot´s of cooperation's with companies in the web 2.0 field - who have massive data amounts too - and a some research in the mobile application area.

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 Edlich] Yes. The impedance mismatch problem is still existing and the root cause of many problems. Most of the companies

I have projects with, store their data in mySQL or Oracle using Hibernate (a de facto standard, but not a standard!). And although I love and teach Hibernate - and I have written a lot about Hibernate - practice shows a total different paradise then promised. There are so many examples.

Lets take a few: First of all, most companies don't use annotations to mark classes and relations on the fly. Why? In huge projects annotations are XML-configurations and they need to be managed somehow. But as there is no annotation management (the intention of annotations is not management), classes and their relations will be configured in hibernate XML configuration files. And this is configuration save, but not refactoring friendly.

But the real world shows so many more problems out there like a defect database scheme. For examples if you have tables without an id or you do reverse engineering or polymorphic queries in these cases you ran into problems. And there are problems in much more areas as stored procedures for Oracle, even correct SQL generation from HSQL or non associative joins.

Object Databases don't have these problems. But of course they have many other problems: the most famous to my opinion is that they do not have the relational model if the problem needs a relational model. So what I am missing in discussion and education is to teach when to use classic relational databases or a relational modes and when something different is better.

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 Edlich] Quite often I use Object Databases and many RAM close solutions, like Prevayler because I think the RAM and cloud/grid space will grow fast. But my most of my industry projects use solutions as Hibernate or iBatis of which I have much experience with.

The lessons I learned together with my team is that:
First: you need to evaluate the database or the mapper better.

But secondly: no matter what you choose, there is a set of problems you always might ran into. Like reporting for ODBMS or the problems I mentioned above for the famous relational solutions. But none really builds up a pro / con matrix. Everyone chooses the standard solutions and is happy to tell the management that everyone does.

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

[Stefan Edlich] As mentioned before, they solve a specific class of problems. If you wish no mapping, performance due to deep object-graphs, easy schema changes, or a different administration, then object databases will make you very happy. But of course if you need a neutral representation of your data with different views and projects coding against it, if you need a sophisticated reporting, specific clustering out-of-the-box, or relational specific DB Services / Tools or Administration, then the classic solutions as Hibernate/mySQL will make you happy. But unfortunately the Hibernate/mySQL solution seems to be the perfect solution because everyone does it. But I have seen quite a lot projects where the classic solution is a total wrong hammer for the current problem.

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

[Stefan Edlich] There are three areas where I would like to see improvement in this time frame. My answers cover only wishes for object databases:

First: To me it looks like there are many ideas outside to help object databases. And if these solutions - as the Stack Based Architecture and the Stack Based Query Language (http://www.sbql.pl) - prove to be superior to strong competition as LINQ, the community and the ODBTWG should do the following: push this standard, someone should provide a reference implementation and we should promote this standard as being superior to LINQ!
Of course many might say that LINQ is the 'standard' so it's great to implement this too. But if we can do better we should implement so and convince the users with marketing and simple examples.

Secondly: I would love to see Java closures. This would give Object Persistence a great push towards the best queries ever.

And third: I would like to see the dynamic languages help Object Persistence more empathically. These new languages offer superior reflection and features for internal DSLs that are perfect to push smart queries in object persistence. As an example: why can't Groovy fire LINQ or AOQL queries and thus help Java?
So these query related point would help in my opinion to let powerful queries sink into each favorite language.