ODBMS.ORG User Report No. 1/08

Editor Roberto V. Zicari- ODBMS.ORG *www.odbms.org*

July 2008.

Category: **Industry**
Domain: **Automation System Solutions for Postal Processes.**
User Name: **Gerd Klevesaat**
Title: Software Architect
Organization: **- Siemens AG- Industry Sector, Germany**

Gerd Klevesaat, starting with C(++) and Assembler development 14 years ago. Consultant, Developer and Architect for COM (ActiveX),.NET (MCSD) and Java. Today working as a Software Architect with focus on J2EE for Siemens AG in Konstanz.

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

Gerd Klevesaat: Siemens provides complete automation system solutions for postal processes.
It covers hardware (conveyor systems, sorting machines, container storage systems) and software (control software, production planning, revenue protection). The software covers a wide range of different subsystems, from low level access to devices and production lines to high level data processing like workflow handling and production planning. Each subsystem has various requirements to store and retrieve data.

My application domain is focused on production control systems to drive the postal processes and its mail streams.
Postal processes and mail streams are very customer specific. This also means that application system development is determined by individual customer requirements and rules.
As a software architect, I am responsible in defining the technological and structural base to drive these processes.

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?

Gerd Klevesaat: Definitely, we have one as long as there is an explicit mapping of objects to the way they are stored. Bridging that gap is tedious and is both a cost and risk factor in the project lifecycle.

Depending on the size of the domain model o-r-mapping can be quire complex. If done wrong, it can be counter-productive in terms of performance and efficiency. Mappings with excessive query statements with a lot of joins are possible. It is a cross-cutting concern that blocks developers to focus on the business problem to solve.

In addition to provide a proper mapping, you are forced to define a query in a special query language. It would be beneficial to use the programming language to leverage compile time checking of query statements and navigation capabilities. There are promising approaches, for example, with LINQ in .NET, native queries in db4o or DataSets in Groovy that deal with this topic.

With db4o native queries you are able to leverage capabilities provided by the programming language. In Java, you would do:

```java
ObjectSet<Person> persons =
    objectContainer.query( new Predicate<Person>() {
                public boolean match(Person p) {
                    return p.getAge() > 42;
                }
            });
```

Your query is defined against properties of the object and operators of the programming language. That is an intuitive approach, especially when accessing object databases. This stuff gets a little bit harder by adding a comparator object to support sorting.

LINQ goes one step further; queries to a database are really language-integrated artifacts. Working with data becomes as natural as working with other language features such as calling a getter to access a property value.

```csharp
IEnumerable<Person> query = from p in persons
                where p.Age > 42
                orderby p.LastName
                select p;
foreach (Person person in query) {
    Console.WriteLine("Name = {0}",person.LastName);
}
```

Persons may transparently be a collection, an array or a table of a SQL Server database. It is extensible. And you use keywords integrated into the language to define the query. However, LINQ is provided for .NET, it would be fine to see something similar for Java.

Groovy has a similar concept which is strongly driven by its dynamic nature and closures. GroovySql provides DataSets to simplify JDBC programming.

```
def ds = sql.dataSet( "persons" )
def persons = ds.findAll { it.age > 42 }
persons.each {
    println "Name = ${it.lastName}"
}
```

This is a very compact syntax.

*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)?*

Gerd Klevesaat: We have enterprise level requirements for persistence and use a clustered Oracle 10g as a database backbone. Mapping of objects and their fields to tables and columns is done using Hibernate 3.

Experiences:
Oracle has proven as a secure and scalable database solution. However, we noticed (performance) problems when data changes frequently due to full table scans and low optimized caching. Oracle provides a good tool support to monitor and identify problems.
Hibernate definitely eases to bridge persistence of objects into the relational oracle database.

However, there were several lesson learned in using O-R-mapping technologies.
1.) Experienced OR-Mapping Know-How was required.

Good and efficient mapping is not trivial. Decisions made on the way of mapping have impact on the model of your domain classes. Model and mapping strategy has impact on the efficiency and performance of the database management system. Knowledge is essential to get this optimized.

2.) Think twice about your domain model when using o-r-mapping. There are issues with class hierarchies and their mapping to a relational database.
A proper domain model from fitting from the object oriented point of view may not fit when used with a relational database.
Say, you have a rich domain model with deep class hierarchy. Each domain class is mapped to its own table (table-per-class mapping strategy). In that case, you have to be careful to avoid a too generic access using references to base classes which may lead to huge sql statements with a lot of joins. Lessons learnt: keep class hierarchies slim and use objects with their concrete type.

3.) Maintenance of mappings
Maintaining a lot of domain objects and their mapping to a relational database can be cumbersome.

Developers have to deal with database mapping issues but, in general, they are not necessarily database experts. You still have to bring database experts and business development experts together. Hibernate saves time in persisting objects. However, some of the time saved is eaten up by issues caused by inappropriate mapping.

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

Gerd Klevesaat: Applying the object oriented development paradigm is complete only by using object persistence. Mapping shouldn't be an issue at all.
Objects are used, passed around, even exchanged when distributed and, more or less, all is handled transparently for me.
It should be the same with storing, querying or retrieving objects. Developers should be able to pass the object to the database system and the database system performs all necessary tasks to persist it as efficient as possible.
It should be the responsibility of the database management system to cache or store it in a file, tables or wherever efficiently. The same

applies to object retrieval. It should be done by the database system who knows how to handle this task.

Object Database Systems are doing exactly that and so, yes, they are a suitable solution from a technical point of view.
If existing relational database systems are required, at least, they should provide a proper interfacing to object oriented development.

Relational databases have strongly established because of standards and tooling. You can freely choose on the query tools by using a SQL editor you like. You can access any database with an established standard API, like JDBC. This makes it possible to use different scaled databases for development and production environments, for unit testing and system tests.
In the field of enterprise applications, relational databases provide tight integration support into application servers. Object databases are still not that well prepared although they have improved. But it is hard to break the relational dominance in the enterprise application area.

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

Gerd Klevesaat: Standardization is very important. Standardization on APIs is the base to provide and improve tools working on these APIs.
Looking at relational databases, you can freely choose your tool to connect to any database as long it provides access using SQL.
I wish to be able and use one tool to connect to any object database using a standard API.

Working on databases has to be as natural as possible for a developer. Developers should be able to access and query a database using one language that he is also using to solve a business problem with.
Db4o has gained a lot of acceptance because of a simple and easy to use Java API.
Simply hand over the object to the set method of an object container representing the database. That's all. The rest is done by the storage system.

With native queries I can write queries leveraging the capabilities of the programming language, e.g. Java. I use its operators against the properties of the object. That is nice and straight forward. It has been enjoyable working with object persistence that way. No mapping, no annotations.

I would like to see concepts like LINQ realized also for Java. Recently, Carl Rosenberger from db4objects has started such an initiative and wish that this project gets momentum.