ODBMS.ORG User Report No. 25/08

Editor Roberto V. Zicari- ODBMS.ORG www.odbms.org
November 2008.

Category: **Industry**
Domain: **Logistics**
User Name: **Martin F. Kraft**
Title: ***Application Architect***
Organization: **Shipping Company** (not disclosed)

Starting off with C, C++, Assembler and Pascal 25 years ago, Martin F. Kraft is now working as an Application Architect overseeing design and actively contributing to the development of an enterprise logistics application using several OO technologies.

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

Martin F. Kraft: Implementing the back office application for a shipping company in GemStone/S, VisualWorks, Java, C++, VB, with thousands of classes and a 10-digit of its instances in GemStone alone, my current role is in designing and/or reviewing application changes as framework team lead, and implementing the more difficult parts together with my small team of software specialists.

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?

Martin F. Kraft: Comment: There is an impedance mismatch between OO and relational models, but when using an ODBMS as intermediate operational store it becomes manageable.

No and Yes. The centerpiece of our Shipping Application Suite uses not only GemStone/S as primary/operational persistent object store but also as an application server, our VisualWorks clients are seamlessly integrated with no mismatch in the object model between client and server.

Nevertheless, we do map the OO persistent data to relational databases in close-to real-time and experience two kinds of issues:

- Object relationships that cannot be mapped one-to-one introduced artificial OO helper-classes that are not persistent. They either represent calculated-on-the-fly information mapped to relational tables and/or columns, or information that represents the relation itself or function as a trigger for downstream apps. Those temporary objects map themselves with a non persistent identity in GemStone and hence cannot be re-synchronized easily.

- Downstream systems that use this relational representation rebuild their OO model not only in a different way than the source system, but also require lots of table joints, where it is sometimes a challenge to reduce table scans and especially the number of round-trips to the RDBMS (e.g. with JBOSS and WebLogic).

For new development, where feasible, we use Web Services between GemStone and Java using the same object model instead of an intermediate relational storage (service oriented architecture).

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)?

Martin F. Kraft: Our central piece uses GemStone/S for managing persistence objects while functioning as our application server and it works great!

Downstream OO systems use o-r-mappings with Hibernate and other technologies, but are much smaller in scale, see Q2.

Besides GemStone/S, I have also used ObjectStore/C++ for Smalltalk (OSST2) as ODBMS, as well as Hibernate/J with RDBMS and Versant/S as ODBMS, and found that concurrency and dynamic typing issues were the most prevalent, besides the architectural limits of such solutions not being able to perform server-side OO logic that not necessarily consists of data retrieval or manipulation – if your needs are less demanding than ours, those ODBMS might work for you just fine.

I also used o-r-mappings with relational databases like RDB7 (with GemStone), Oracle (with VisualWorks and C++), Sybase (with Java), and DB2 (with Java, VisualWorks, VisualAge, C++).
The mappings were either all self-made without any common API (tools like TopLink still require manual mapping configuration and it is up to a very skilled developer to do a reasonable job), or in the case of true J2EE it still seemed far too slow compared to ODBMS implementations.

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

Martin F. Kraft: Yes since it is a natural relationship, I would vote for GemStone/S being the reference implementation.

Nevertheless, RDBMS are mainstream and more easy to use by developers not as skilled as required for OO databases and application servers, therefore the efforts to support mainstream RDBMS as OO storage are well aimed. Eventually those o-r-mapping solutions should become almost as good as ODBMS.

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

Martin F. Kraft: Regardless of struggles due to the o-r-mapping issues caused by the natural difference between OO and relational models, I would like to see performance improvements within J2EE as done with GemFire/J, e.g. in OC4J and many others.