ODBMS.ORG User Report No.8/08

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

July 2008.

Category: **Industry**
Domain: Financial Services
User Name: **John Davies**
Title: Technical Director
Organization: ***Iona*, UK**

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

John Davies: I have worked my way up through the enterprise, first in hardware then Assembler, C, C++ and finally Java languages. Since the very early 80s I worked in investment banking and from the late 90s I've had roles from head of trading technology to Global Head of Architecture at JP Morgan Chase. I started a company in 2000 called C24 selling SWIFT integration solutions and that was sold to Iona in March 2007 and I'm now the Technical Director of Iona's Financial Services division. Most of my work today is advising to large organizations on high performance and low latency enterprise architectures.

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?

John Davies: This "impedance mismatch" is a serious problem in the enterprise, upwards of 25-33% of development time is being wasted squeezing objects into relational persistence, usually termed Object Relational Mapping (ORM). While the examples of such tools demonstrate the simplicity of ORM, real life is usually several orders of magnitude more complex and the whole idea breaks down. Even the best tools create an incredibly inefficient model, resulting in serious performance issues.

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)?

John Davies: It is not the objects that we are trying to store but the data held within them, their state. It's rare that an object hierarchy is so complex that we have problems persisting their state, however in the case where the objects have been bound to a message or relate directly to a message, typically XML it is really the XML that we are trying to persist and not the objects. Certainly, in the banking world and from what I can see in other verticals, telco and government for example, this is a serious issue. Several banks are investing serious money into developing solutions around this problem. Again I have to emphasize that this is hierarchical message persistence not necessarily object-persistence, this is usually written with object-oriented languages, predominately Java, so a side-effect would be a Java or object persistence mechanism.

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

John Davies: These days we have enough memory on average servers to store hundreds of gigabytes of data/objects.
Several companies entire business model revolve(d) around "in memory" persistence, Tangosol (recently acquired by Oracle) created distributed maps, GigaSpaces will host huge object collections using JavaSpaces. In many cases we don't even need to touch a disk and you could technically refer to these as in-memory object databases. Now if you actually need to write the data to disk then you're back to the impedance mismatch problem, again the most frequent use-case I've seen outside of ORM is writing indexed database BLOBS.

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

John Davies: Object Databases seem to have very little footprint in my experience, mainly because they have proprietary interfaces, what we need is an object persistence API that all vendors adopt, the nearest we got to this was the Java map interface used by Tangosol and that was a huge success. The problem is hierarchical persistence not object persistence, when the vendors provide non-proprietary solutions for this I think we'll see huge adoption.