ODBMS.ORG User Report No. 17/08
Editor Roberto V. Zicari- ODBMS.ORG www.odbms.org
August 2008.

Category: **Academia**
Domain: *Research/Education*
User Name: **Mohammed J. Zaki**
Title: *Associate Professor*
Organization: **Rensselaer Polytechnic Institute, USA**.

I am an Associate Professor of Computer Science at RPI. I received my Ph.D. degree in computer science from the University of Rochester in 1998. My research interests focus on developing novel data mining techniques, especially in bioinformatics and web mining. I have over 170 publications on data mining, I am an associate editor of the leading journals in the field (e.g., ACM TKDD, IEEE TKDE, DMKD, SAM), and I am the program co-chair for SDM'08 and SIGKDD'09.

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

Mohammed J. Zaki: I am interested in large-scale data mining over massive datasets. In particular I have focused predominantly on mining complex patterns (sets, sequences, trees, and graphs), with applications in bioinformatics, and web mining.

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?

Mohammed J. Zaki: I have been struggling with the impedance mismatch problem from the very beginning, since the datasets I deal with invariable do not fit in memory, and moreover represent complex objects (e.g., graphs, protein structures, gene expressions, web logs, etc.). I do most of the development work in C++, and I have yet to find an elegant/scalable solution to the "persistency" problem.

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)?

Mohammed J. Zaki: Let me elaborate a bit on the use of the term persistency. The most common usage seems to be a way to "serialize" the objects to disk, and retrieve them as needed, or when queried. OODBMS have solved this problem to a large extent (depending on your specific needs), but there are still problems, and more so if you want to use open-source systems.

My problem is with the second notion of persistency, namely the ability to transparently deal with objects without having to worry whether they are in memory or on disk. For example, if I am writing "generic" graph mining or graph traversal code, say in C++ (e.g., using STL), then I would like to write generic algorithms that work automatically over in-memory or out-of-core graph datasets. The system should automatically bring in objects from disk on demand, with all the great caching/buffering solutions built in, without having to explicitly serialize/deserialize the objects.

Specifically for C++, there are some efforts that add persistence to STL, such as PSTL (which uses mmaped-files) and STXXL, but the problem is far from being solved in a general way. In the Data Mining Template Library (DMTL; http://sourceforge.net/projects/dmtl/) open-source toolkit, we ended up using the serialize/deserialize approach for supporting large out-of-core datasets, as well as for storing the intermediate pattern objects created during the data mining process.

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

Mohammed J. Zaki: OODBMS are suitable for the first form of persistency, but not for the seamless, transparent persistency for complex object querying, indexing, and mining.

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

Mohammed J. Zaki: I would like more open-source efforts on transparent memory/disk management, with special focus on scalability/performance. In essence, I would like to make persistency a first-class feature of the programming language of choice.