

DATABASE

TRENDS AND APPLICATIONS

Solutions for the Information Project Team • www.dbta.com

Volume 20, Number 7 • July 2006

APPLICATIONS INSIGHT



GUY HARRISON

Does Microsoft Want to Replace SQL?

Since time immemorial - well, the early 1990s anyway - object-oriented programmers have felt oppressed by the need to work with relational databases. To many object-oriented (OO) programmers, the relational database is a cumbersome dinosaur that is at odds with the elegant OO model. While the RDBMS represents data as a set of two dimensional tables, an object-oriented application represents the same data in objects that typically mix data from multiple relational tables. Constructing and deconstructing objects from relational data is a source of pain to OO applications.

In the early stages of the OO revolution, many developers believed that the relational database would give way to a new object-oriented database, the OODBMS. In my view, OODBMS advocates forgot the major lesson of the RDBMS revolution: any database systems that prevent non-programmers from getting to that data will not win business support. The relational data model, and the ecosystems of BI tooling (including the humble Excel) provide business value beyond that of the

applications that create and maintain that data. More recently, attempts have been made to marry the RDBMS and the OO worlds through Object-Relational-Mapping (ORM) schemes. An ORM provides the OO programmer with an object-oriented view of the RDBMS data. The mapping is typically contained within XML configuration documents. J2EE provides a standard ORM in the form of Enterprise JavaBeans (EJB), though open source ORM solutions such as Hibernate and IBATIS are more popular.

Use of ORM has not been so widespread in the .NET world, but this may soon change with the release of Microsoft's "ORM and beyond" offering, LINQ (Language Integrated Query). LINQ provides an ORM foundation not dissimilar to Hibernate, but goes beyond Java ORM systems by providing a data query and manipulation syntax that can be used to manipulate mapped objects (e.g., relational data) or other data sources (such as XML documents). This approach allows the developer to perform SQL-type operations without having to embed and process

SQL strings within .NET code. It is not hard to create examples in which queries can be expressed more elegantly in LINQ than in C# or Java.

I was initially quite intrigued by the LINQ approach: certainly all the mechanisms for issuing SQL in Java or C# feel messy and unnatural, so a native query capability seems attractive. However, my initial investigations of LINQ left me feeling disappointed. The "easy" cases were hard to implement, the code generation is currently awkward, and only SQL Server is supported. Lastly, I did not find the LINQ syntax intuitive.

Microsoft will undoubtedly remedy LINQ's current deficiencies. At the moment LINQ is not expected to become production-ready until .NET 3.0. However, while LINQ certainly bears watching, I doubt that it will become the successor to SQL.

Guy Harrison is chief architect for database tools at Quest Software and has specialized in database administration, development, and performance tuning. He is the author of "Oracle SQL High Performance Tuning" (Prentice Hall).