Craig S. Mullins

Return to Home Page

April 2001

The Managers Publication of Data Solutions

DATABASE TRENDS

Why Data Still Matters

By Craig S. Mullins

Over the years, many technologies and marketers have claimed that data has become irrelevant and that some “new and improved” technology, technique, or ideology will replace data as the center of IT and data processing. But it has yet to happen, and it never will happen, either!

One pretender to the throne was (dare I say, is) object oriented technology. The object proponents claim that objects, because they encapsulate both data and the processes that manipulate the data, are superior to data. This is pure fantasy. OO development works well for pre-planned, recurring workloads. When the developer can plan and implement **all** of the methods required for the object, the OO way of doing things will work fine. But is this really anyone’s idea of reality?

Many workloads and requests are unplanned. Ad hoc queries, OLAP, and data mining require access to data in ways that were not originally devised when the data (or object) was first created. A proper database design, created from a logical data model, and implemented using a relational database enables ad hoc access to data using SQL. If instead you had to rely on the “OO way” you would have to develop new methods for the objects to look at the data encapsulated therein in a different way. The performance of the new methods is likely to be poor if the data within the object must be accessed in very different ways.

But the matter gets further complicated if your unplanned data analysis requires data encapsulated in multiple objects. The OO way would require multiple methods to be invoked by each object that contained the required data and then some program to cobble the results together. The beauty of relational databases using SQL is their flexibility and suitability to perform unplanned data gathering and analysis, quickly and with minimal effort. And without redesigning the database or writing complex new programs (methods).

Furthermore, with today’s relational database technology, code can reside in multiple places: procedural code on the database server in the form of stored procedures and user-defined functions, within active database rules in the form of triggers and constraints, and on multiple application tiers. Rigid conformance to encapsulated methods within objects imposes a strict development methodology that may cause more harm than good.

The argument made by OO proponents goes something like this: First of all, an object is a more natural model for representing the real world. By adhering to OO tenets the potential for reuse is high. Furthermore, OO properties such as inheritance and polymorphism provide additional flexibility for application development, enabling the programmer to modify objects to suit the circumstances. The overall goal is to use OO development techniques to create reusable components that can be combined together to build applications.

The argument is compelling, but misleading. To whom is an object “more natural,” and as compared to what? It is very natural for people to relate to most data the way a relational database does, as rows and columns – we do it every day on business forms, checkbook registers, spreadsheets, phone books, and so on. Furthermore, it is quite possible to create reusable components without adhering to the OO philosophy or using OO development techniques. CBD (component-based development) proponents are doing this very well today without rigidly conforming to OO development rules.

Now, I don’t want to totally and completely dismiss object-oriented technology, but it is within the realm of programming that it provides benefit. Any benefits OO can provide to DBMS technology have already been incorporated into most DBMS products (that is, extensible data types). For OO to achieve any long-term success, a practical OO development environment that interoperates with and maps to relational databases must be established. And it must not compromise good data modeling and relational database design practices in doing so.

The bottom line is that data has intrinsic value and it should not be forced to co-exist with program logic (methods) before it can be stored. Neither should limitations be placed on data that would require complex object-oriented program logic to be written for every data access requirement.

Data is a corporate asset and should be treated as such. There are myriad benefits to modeling data. Persistent data stored in relational databases is an elegant and useful way to support the planned and unplanned needs of your organization. And it will continue to be so for many years to come.

From Database Trends, April 2001.