

# Database Directions

**Paul Grabscheid** is vice president of strategic planning for InterSystems.

**DDJ:** InterSystems provides both general-purpose object databases and database systems specifically for the healthcare industry. What's unique about healthcare that requires specialized database tools?

**PG:** While InterSystems Cache is widely used in healthcare, the characteristics that make it attractive there are increasingly important in other areas as well. First and foremost, the clinical side of healthcare deals with large amounts of unstructured and semi-structured data. The data that comprises your medical record is likely quite different from mine and the nature and format of data collected and stored has evolved over time. All of this argues for the flexibility inherent in an object model, both to capture the richness of data and to effectively support the storage and use of new data types. Increasingly, for instance, genomics information is being captured for diagnostic and treatment purposes. Another challenge of healthcare is to enable graceful evolution of data models. Certain clinical data remains valuable for years or decades, rather than the weeks or months found in other application domains. This requires an ability to add new data, new data types, and new relationships without "breaking" existing applications and structures and without requiring disruptive database reorganizations or unload/reload cycles. Object database technology facilitates this through schema evolution, enabling incremental "nondestructive" changes to database structures, and through the object concept of polymorphism, which provides "safe" access to related but distinct object types or versions.

**DDJ:** In terms of object/relational, impedance mismatch is an issue that just doesn't seem to go away.

**PG:** I think that impedance mismatch is becoming more of an issue, not less. Ten or 15 years ago, object concepts were viewed as powerful but with a steep learning curve. These concepts were foreign to a generation of developers steeped in filesystems and relational databases. Developing an object-based application required more steps and more work than traditional approaches. Today the situation is reversed: Many (perhaps even most) developers think about the problems they are trying to solve and the data models they are using to solve them in object terms. The object concepts of inheritance, encapsulation, and polymorphism have become the natural way to frame solutions and architect applications. In this world, using a relational store at the backend adds extra development steps because of the need to architect and implement an object-to-relational mapping strategy.

The shift from relational to object data models represents a shift to dealing with data complexity at design time, rather than at coding or execution time. As application data models get more complex, the benefit of a rich object model that naturally represents bidirectional relationships, containment and hierarchies, for instance, grows in importance. An object model in the database decreases development effort and reduces runtime overhead compared to the multiple JOINS required in a relational system.

**DDJ:** I'm generally familiar with "transactional databases," but what's "transactional bit-map indexing" all about?

**PG:** Traditionally, there has been a strong separation between transaction processing applications and analytical activities that are carried out using separate data warehouses. This separation served to protect the performance of run-the-business transaction systems and was made possible by the fact that most data warehouse activity involved long-range decisions that did not require up-to-the minute data. Often, weekly or monthly batch updates from transactional applications to the warehouse were sufficient. Bit-map indexing was perfect for the data warehouse approach: It offered excellent performance for complex queries, albeit at a "cost" of poor performance for inserts, updates, and deletes. As long as this cost occurred infrequently and did not impact the transactional systems, everyone was happy.

Now, the focus has shifted to employing Business Intelligence and other analytical technologies as part of transactional applications, in order to support the much larger range of operational decisions that must be made every day. To address this need, we began work on a new

approach to bit-map indexing that combined high performance for queries as well as for updates. From a technical standpoint, this took two forms: Advances in the way bit maps are stored, replacing the simplistic one-bit-per-row approach with an adaptive compression technique that provides substantially more efficient storage, and a more optimized approach to dealing with bit maps in the database engine. The resulting transactional bit-map index technology underlies our new developments in real-time analytics.

**DDJ:** "Database-as-a-Service" is an emerging topic, particularly in terms of Cloud computing. What does this mean for developers? For database vendors?

**PG:** I'm not convinced that Database-as-a-Service will fly as a business concept. While there's continued interest in purchasing certain types of applications on a service basis, the customers I speak with have much less interest in "infrastructure as a service." That said, the concept has some implications for database technology that are positive for a variety of database delivery models. Database-as-a-Service demands a simplicity of system administration that has not been seen in typical enterprise database software. It also requires an ability to add and remove system capacity and transparently shift workloads without application disruption. Improvements in these areas will decrease operating costs and improve total system availability, with benefits to users of database technology regardless of delivery model. **DDJ**

