JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 2, March-April 2003

A Retrospection On Niche Database Technologies

Won Kim, Cyber Database Solutions, Austin, Texas

1 INTRODUCTION

Today commercial relational database management systems (RDBs) and supporting tools (database design tools, graphical database browsers, application generators, query tools, etc.) from several major vendors are used to suit most data management needs around the world. The database market is now dominated by the Big Three: IBM, Microsoft, and Oracle. Initially, commercial RDBs supported the first normal form relational data model and SQL query language, along with basic transaction management and access authorization facilities. They mostly ran on a few UNIX platforms. The scope and sophistication of the functionality they support, and the performance they deliver have all substantially matured during the past twenty years. They have also been made to run on most computers and operating systems. However, the often frustratingly long periods that it took major vendors to shore up various shortcomings of RDBs have led to the introduction of what I will call "niche database" technologies during this period of maturation. Each niche database technology came about to address typically one major shortcoming in then-existing RDBs. The technologies the vendors introduced and the efforts they made to market the technologies often provided a key incentive to major vendors to upgrade their RDBs with the same technologies. The financial wherewithal, customer loyalty, and marketing prowess of the major RDB vendors have simply overpowered vendors of niche database technologies. For these and other reasons, except for about a dozen vendors, niche database technology vendors have generally failed to grow into sustained large business entities. Today the maturity of RDBs, the expectation of the customer base, and the domination of the market by the Big Three make it very difficult for the entry of new niche database technology vendors.

In this article, I will provide a retrospection on niche database technologies by briefly examining them and how they have or have not been adopted by major RDBs. For the purpose of this article, I will classify niche database technologies into five categories: performance engines for certain types of database operations and environments, performance monitoring and tuning tools, extensions to the first normal form relational model, support of non-alphanumeric data types, and data integrity tools. I will not include

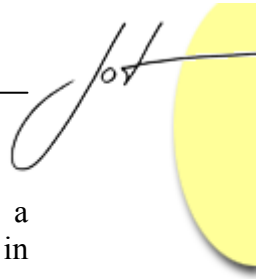
Cite this column as follows: Won Kim: "A Retrospection On Niche Database Technologies", in *Journal of Object Technology*, vol. 2, no. 2, March-April 2003, pp. 35-42.
http://www.jot.fm/issues/issue_2003_03/column4

discussions of various types of supporting database tools, such as query tools/report writers, database design tools, application generators, database browsers, etc. Although these are not end-user applications, I regard these tools as applications of RDBs nonetheless.

2 PERFORMANCE ENGINES

The performance of RDBs has primarily been driven by the demands of Online Transaction Processing (OLTP) applications. Such applications tend to retrieve only a small fraction of a large database, update some, and store them back. The large number of simultaneous transactions has forced the RDBs to run on high-end servers and multiple computers. Performance techniques that have been incorporated into RDBs include access methods, query optimization based on cost estimation, process architectures, optimized transaction management, parameterized performance tuning, parallel bulk loading of data, among others. Access methods include B+-tree-based indexes, hash tables, pre-sorted tables, and segments of disk pages. A table is entirely stored in one segment of disk pages, and each disk page is formatted to allow for quick access to individual records and for quick linkage to another page. The cost-estimation-based query optimization technique computes the estimated costs of all reasonable 'plans' for processing any given query by making use of all access methods available for relevant tables, and selects the lowest-cost plan. The transaction management mechanisms of concurrency control and recovery have been highly optimized. The instruction counts for testing and setting locks on data items have been minimized. Logging of updates to data to secondary storage has been minimized to reduce I/O costs. RDBs provide lots of parameters that database administrators may set in order to optimize performance for their particular application environment. The parameters include the amount of space for a disk segment, the amount of free space to be left in each disk page to allow for future record insertions, the interval for compactifying disk storage by purging records that are marked deleted but not physically deleted, etc. RDBs have incorporated a multi-process and multi-thread structure to fully utilize the CPU cycles while waiting for disk I/O. The process structure is also essential for RDBs running on Symmetric Multi-Processors and clusters of processors both of which have multiple CPUs. Parallel query processing techniques have been incorporated into RDBs wherby a complex query is decomposed into a sequence of subqueries such that some of the subqueries are processed in parallel on separate computers, and subqueries are processed in a pipelined fashion to reduce total response time. RDBs have also adopted parallel processing techniques to bulk load data, back up data, create indexes, etc.

Despite 20-plus years of efforts for performance improvements, there remain at least two special situations that conventional RDBs do not yet cover. These are applications whose databases fit into main memory, and applications that require largely "scan-oriented" access to a database. Main-memory database systems, such as TimesTen, Altibase, Datablitz, Prisma, etc., perform all database operations without incurring I/O, after loading data from secondary storage. These systems have reduced instruction counts



for most operations by removing considerations for access to secondary storage. As a result, main-memory database systems can often exhibit a ten-fold improvement in performance over conventional RDBs. Of course, conventional RDBs running on a larger main memory can deliver higher performance than those running on a smaller main memory, and the advantages enjoyed by main-memory database systems over conventional RDBs can be partially offset by the use of a larger main memory. Main memory database technology has an interesting potential when operating systems support a 64-bit architecture, which will make main memory very large indeed, and when the cost of main memory becomes much lower. The challenges facing main-memory database systems also include the need to support most of the SQL query language, and the resistance from system administrators who do not care to administer yet another database management system beyond the conventional RDB that they most likely already administer.

"Scan-oriented" access to a database is one that has to scan an entire table (or database), possibly more than once to perform an operation. Examples of a scan-oriented access include queries that require all records of a table to be grouped by a certain column order, queries that compute aggregation functions on certain columns of a table, queries that require a table to be pre-sorted, queries whose selectivity is low, operations that restructure a table, etc. A grouping query, for example, groups all Persons by their Age. An aggregation query, for example, computes the Average Salary of all Persons. A low-selectivity query, for example, is one that selects all non-smoking Persons. Table restructuring operations include, for example, decomposing a column into two or more columns (e.g., a single Address column into StreetAddress, City, and State columns), and changing the continuous numerical values in a column (e.g., Salary of $55,400) into categorical values (e.g. Salary category of $50,000 to $59,999).

Such access methods as indexes and hash tables have been very useful in evaluating queries for OLTP applications. However, they are useless in evaluating scan-oriented queries. Online Analytical Processing (OLAP) applications are those that tend to "analyze" a large amount of data, rather than retrieve only a small number of records from a large database. Analyses of data tend to require summarization and aggregation of data along multiple dimensions (roughly, columns); for example, total sales of a certain model of computer by Region, by Month, by Store, etc. As such, they tend to require an entire table to be retrieved and examined. Such software products as MaxScan and Sybase ASIQ have been designed to perform scan-oriented queries fast, and are up to 100 times, and on average 10 to 20 times faster than conventional RDBs. MaxScan uses a special storage layout scheme, data compression scheme, and hashing techniques to gain high performance. Sybase ASIQ resorts to the encoding of all data in tables in order to reduce the size of the tables and to expedite query evaluation. Despite compelling performance advantages, these products face the same challenges mentioned earlier that main-memory database systems face.

The need to summarize data along many dimensions by OLAP applications led to M-OLAP systems, such as IRI Express (acquired by Oracle), Arbor Essbase (acquired by Hyperion), and Pilot Decision Support System (acquired by Accrue). M-OLAP systems

are OLAP servers that pre-compute summaries of data along all possible dimensions, and store them in a storage system called a multidimensional database system. Since the summaries are stored, queries return answers rapidly. However, the pre-computation of the summaries is in general very time-consuming, and the storage needed for the summaries, unless null results are excluded, often far exceeds the original data. Further, updates to the original data invalidate the summaries and require re-computation. Unlike M-OLAP systems that require a special storage system, R-OLAP systems were offered by vendors such as MicroStrategy, Cognos, etc. R-OLAP systems use an RDB as storage system, but is generally much slower than M-OLAP systems. Microsoft offers both M-OLAP and R-OLAP systems as Analysis Service to its SQL Server RDB.

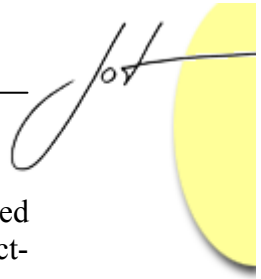
3 PERFORMANCE MONITORING AND TUNING TOOLS

Given the complexity of today's RDBs, the operating systems on which they run, the applications that run on RDBs, the size of the databases, and the computers on which the RDBs run, it is exceedingly difficult for database administrators to ensure optimal performance of RDBs and applications. SQL queries, database buffer management, disk I/Os, database contention (locks, latches, wait events), and even the operating system (CPU, I/O, and memory utilization) can all adversely impact performance. There are several vendors, besides the major RDB vendors, that offer tools for aiding database administrators in monitoring and tuning the performance of RDBs. They include Computer Associates, BMC Software, Embarcadero, Quest Software, etc.

4 EXTENSIONS TO THE FIRST NORMAL FORM RELATIONAL MODEL

Over the past twenty years, many people have complained about the first normal form relational model supported in conventional RDBs. The first normal form relational model basically says that each column in each record contains atomic data. Atomic data is data that is stored and retrieved as a single unbreakable unit. This is rather unnatural for managing sets of data or data that can be described in further detail. For example, in the Children column of a Person table, the user may store the names of two children (Tom, Dick); but RDBs treat this as a single data item, rather than two separate data items. Further, in the Hobby column of a Person table, the user may store detailed information about a Person's Hobby, for example, as (name: tennis, years-played: 5, hours-spent-a-month: 10). Again, RDBs treat the entire Hobby data as a single unit, not understanding the constituent data elements.

The need to naturally model and query the nested structure of some data, such as Hobby as shown above, led to non first-normal form RDBs, such as UniData (acquired by Ascential Software).



The limitations of the first normal form relational data model came into a renewed focus when during the 1980s various vendors and researchers tried to create object-oriented database systems (OODBs). Initially, the term object-oriented database system caused a great deal of confusion, especially to the RDB community. What early vendors such as Mosaic (which was renamed Ontos), ObjectDesign (now eXcelon), and Servio Logic meant by an OODB was merely a persistent object-oriented programming language system. Ontos and ObjectDesign wanted to offer a persistent C++ programming language system, while Servio Logic wanted to offer a persistent Smalltalk programming language system. As such, their initial focus was simply to store C++ or Smalltalk objects in a "database", have users retrieve them by providing pointers to those objects, and manipulate the retrieved objects in main memory. Objects are data elements and methods, along with the inheritance relationship on classes. The original conception of an OODB was alien to the RDB community which has been trained to think of a database system as providing a SQL-based query language and transaction management facilities. This led to a period of rather confused exchanges between the OODB camp and the RDB camp. The RDB camp charged that OODBs are not database systems because they do not support ad-hoc queries, and force users to pointer-based navigation of a database, something that was eliminated when database technology transitioned from the hierarchical databases era to the RDB era. The OODB camp countered by saying that it makes no sense to issue awkward SQL queries when pointer-based navigation among objects in main memory gives so much higher performance. Later, vendors of OODBs, both the early ones and new comers like O2, added ad-hoc query languages and transaction management, making them more like traditional database systems, while still retaining the pointer-based navigation of memory-resident objects.

In the meantime, UniSQL, followed by Illustra, saw the futility of the debates between the two camps and found a way to extend the first normal form relational model to an object-oriented model as embodied in object-oriented programming languages. In the early 1990s they came out with object-relational database systems (ORDBs). The object-oriented model includes the nesting of classes, unique object identifiers, methods attached to a class, inheritance of attributes and methods along a hierarchy of classes, and the generalization/specialization relationship among classes along a hierarchy of classes.

Illustra was later acquired by Informix, which in turn was acquired by IBM. The Illustra technology was integrated into Informix RDB. The UniSQL technology was later distributed to several different corporations in the US and the Far East. The trail that UniSQL and Illustra blazed in the 1990s led to the eventual extension of SQL to SQL3 by adding object extensions to SQL, and incorporation of the object-oriented model in Oracle 9i and DB2 8.1. It is likely that other major RDBs will also directly support SQL3 in the foreseeable future, thereby transitioning to ORDBs.

Further, RDB vendors now support persistence for object-oriented programming languages, including C++ and Java. Basically, they provide a wrapper between object-oriented programs and ORDBs, negotiating the differences between the object-oriented model of an object-oriented programming language and that of an ORDB.

One key challenge facing ORDBs is the need to have most of the database tools that have been designed to work with RDBs. Such tools, including graphical browsers and application generators, etc., do not understand the object-oriented model, and need to undergo a substantial overhaul to bring the object model, such as the nested table and inheritance, to the users.

Beyond objects, support for temporal data has been a topic of research for a long time. Temporal data includes historical data and time series data, and the notions of transaction time and valid time. Transaction time is the time when data is entered, and valid time is when the data is valid. There have been several prototypes, such as T Helper, ORES, Tzolkin; and proposals for extending SQL, such as SQL/Temporal, TSQL, TLSQL, etc. However, native support for temporal data has not made its way into major RDBs. A few major RDBs support temporal data by means of data blades.

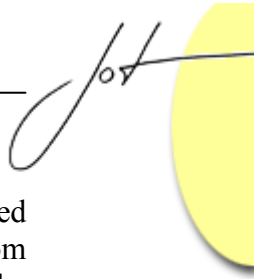
5 SUPPORT FOR NON-ALPHANUMERIC DATA TYPES

Conventional RDBs have been originally designed to support primarily alphanumeric data types. They have later incorporated support for binary large objects (BLOBs), and character large objects (CLOBs). Illustra introduced the notion of plugging into RDBs software modules that implement support for specific non-alphanumeric data types. Illustra called such software modules "data blades". Oracle followed suit with "data cartridges", and IBM added "database extenders" to DB2. These software modules define and implement a particular data type and various operators on the data type.

Although the "blades" represented a key step toward adding support for non-alphanumeric data types, they do not bring the full powers of an RDB to bear on the non-alphanumeric data types, such as an integrated query language, query optimization and processing, transaction management, etc.

Several niche vendors have extended RDB engines by adding native support for spatial data types and spatial operators. Spatial data types include a point, line, rectangle, polygon, polyline, circle, etc. Spatial operators include overlap, abut, contain, above, below, left of, right of, nearest, etc. To support the evaluation of spatial queries, that is, queries that contain spatial data and spatial operators, they have typically implemented an indexing mechanism based on an R* tree. KT Data has extended the UniSQL object-relational database engine with spatial data types and operators. Paradise and Monet support spatial data types and operators, along with parallel query processing techniques. Intergraph offers MGE-GeoData Manager for managing spatio-temporal data. Oracle has also integrated support for spatial data types and spatial operators in its RDB. These are more ambitious systems than RDBs that support spatial data types as blades and coordinate the query processing by a conventional RDB and a separate spatial data blade.

XML (eXtensible Markup Language) is another non-alphanumeric data type that has attracted a lot of attention. During the past several years, XML has become a standard for expressing a wide variety of data that lend itself to a hierarchical structure. The nested structure of XML maps naturally to the aggregation hierarchy of an object model.



Therefore, such vendors as ObjectStore/eXcelon and Poet have used their object-oriented database systems to support the storage and querying of XML data. BADA IV from ETRI, Korea, also provides native support for XML. Major RDBs initially supported the XML data type as blades. However, now Oracle provides native support for XML by making use of the nested table facilities.

Some major RDBs have also added support for some multimedia data types, including images and video. Informix, DB2, and Oracle provide data blades for these data types and operations on them. In the data blades, they have adopted multimedia contents recognition and indexing technologies provided by vendors such as Excalibur (acquired by Convera) and Virage."

6 DATA INTEGRITY TOOLS

Although RDBs provide various mechanisms to ensure integrity of stored data, even today have fall well short of ensuring data integrity. The data-integrity mechanisms RDBs directly support include transaction management, data type constraints, and integrity constraints. Transaction management includes control over concurrent accesses to the same data by multiple users (or transactions), recovery after both soft crashes (that corrupt data in memory) and hard crashes (that corrupt data on secondary storage). Data type constraints are restrictions to user-specified data types for the values in columns, such as integer, character string, etc. Users may specify integrity constraints on certain columns of a table. These include a primary key – foreign key constraint on columns of two tables, the UNIQUE constraint on a primary key column, the NULL constraint that allows a null value in a column.

Data type constraints are not adequate to prevent wrong data from entering the database. For example, an integer value range data type constraint [18..65] can prevent integers outside the range from being stored, say, in the Age column of a Person table. However, an RDB cannot know if the number 25 being stored is a wrong Age for a Person. An integer data type constraint cannot tell the measurement unit of the number 125 being stored, say for the Weight of a Person. Similarly, a string data type constraint cannot prevent misspelled words from being stored, say, in the Name column of a Person table. Further, because of the overhead involved in checking constraints or because of carelessness on the part of the users, appropriate data type and integrity constraints are often not specified. For example, rather than specifying a value range [18..65] for the Age column of a Person, the user may specify Integer, thus allowing any integer value for the Age column. Further, a constraint such as “the average Salary of all Persons should not be more than 30% lower than the highest Salary” will require computing (or at least looking up) the average Salary every time the Salary of a Person is updated. As such, such a “complex and expensive” constraint is often not defined.

The importance of having correct data, and the realization that RDBs do not really guarantee correctness of data have led to several “data quality” tools, from vendors such as First Logic, Trillium Software, Vality/Ascential Software, etc. These tools help the

users to detect wrong data, often names and addresses of people using directories of names and addresses, and correct them. There is a continuing need to periodically test quality of data in enterprise databases, but often this is ignored. This is one niche database technology that appears to be open to niche vendors.

7 CONCLUDING REMARKS

Today there are various niche database vendors, offering data integrity tools, database performance boosters, main-memory database systems, database performance monitoring tools, spatial data managers, temporal data managers, etc. Major RDBs offer rich functionality and are supported by a wide variety of tools, and run on most platforms, and customers expect even niche database engine products to match the functionality and platforms of the major RDBs. As a result, it is becoming ever more difficult for vendors to create a viable business offering niche technologies involving a database engine, and even if it is possible, the window of opportunity is limited to a few to several years, before major RDB vendors incorporate such technologies in their RDBs.

Further, during the past twenty years, customers of RDBs have shown that certain things are not of great importance, including expressibility (ease of coding) in the application programming interface, and less than spectacular (less than 2-3 times) performance advantage. In other words, niche database technologies whose purpose in life fall into these do not have much chance of supporting a viable business.

In a future article, I will explore some possible niche database technologies that are still needed and that may support a viable business model for vendors.

About the author

Won Kim is President and CEO of Cyber Database Solutions (www.cyberdb.com) and MaxScan (www.maxscan.com) in Austin, Texas, USA. He is also Dean of Ewha Institute of Science and Technology, Ewha Women's University, Seoul. Korea. He is Editor-in-Chief of ACM Transactions on Internet Technology (www.acm.org/toit), and Chair of ACM Special Interest Group on Knowledge Discovery and Data Mining (www.acm.org/sigkdd). He is the recipient of the ACM 2001 Distinguished Service Award.