DATABASE
TRENDS AND APPLICATIONS
Solutions for the Information Project Team • www.dbta.com
Volume 22, Number 2 • June 2008

Is the Next DBMS Revolution Looming?

By Guy Harrison

For the first time in over 20 years, there appear to be cracks forming in the relational model's dominance of the database management systems market. For the first time, credible alternatives to the relational database are emerging. While it would be reckless to predict the demise of the relational database, it is feasible to imagine the relational database as just one of several choices for data storage in next-generation applications.

DBMS Revolutions

Relational database architecture became dominant throughout the 1980s in conjunction with the rise of client-server architectures. The relational database made it easier for business to access and leverage data. Yet, from an application developer's point of view, the relational model was not ideal. The relational database management system (RDBMS) came to prominence during the same period as object-oriented (OO) programming. While the relational database represented data as a set of tables, OO represented data in objects that not only associated behaviors, but which also had complex internal structures. The disconnect between the two created an "impendence mismatch" that reduced application cohesiveness.

To resolve this disconnect, the object-oriented database management system (OODBMS) was established. In an OODBMS, application data is represented by persistent objects that match the objects used in the programming language. However, the OO model was programmer-centric and did not address business intelligence needs. The industry standardized on the RDBMS. As a workaround, many application frameworks developed object-relational mapping (ORM) schemes which allowed object-oriented access to relational data.

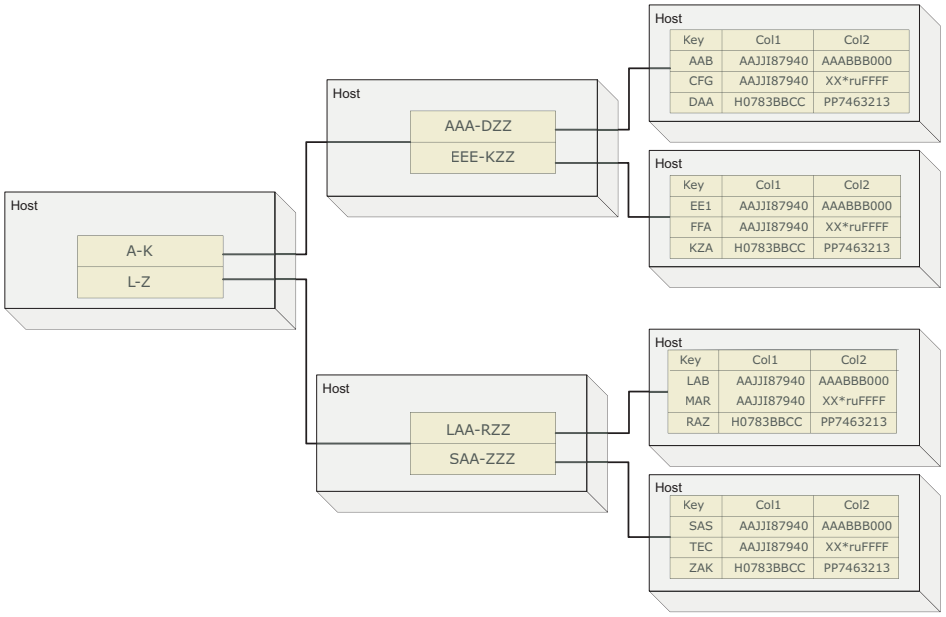

Figure 1: Cloud databases distribute data across many hosts

Utility Computing

Since 2000, the IT industry had been subjected to unrelenting pressure to reduce cost. It had been clear for some time that the utilization of computing resources was inefficient. Because each application used dedicated hardware, the hardware had to be sized to match peak application processing requirements. Off-peak, these resources were wasted.

Utility computing introduced the idea of allowing computing resources to be allocated on demand in much the same way a power company makes make electricity available. It could reduce cost through economies of scale and by averaging out peak demands between applications.

Virtualization, grid computing and the Internet as a universal wide area network have combined to deliver an emerging realization of the utility vision as a computing "cloud." In a cloud computing configuration, application resources - or the application itself - are made available from virtualized resources located somewhere in the Internet (e.g., in the cloud).

RDBMS Challenges

Most components of modern applications can be deployed to a virtualized or grid environment without significant disruption. Web servers and applications servers all cluster naturally and resources can be added or removed

from these layers simply. Unfortunately, it's much harder to cluster databases. In a traditional database cluster, data must either be replicated across the cluster members, or partitioned between them. In either case, adding a machine to the cluster requires data to be copied or moved to the new node. Since this data shipping is a time-consuming and expensive process, databases are unable to be dynamically and efficiently provisioned on demand.

Oracle's attempt to build a grid database - Oracle Real Application Cluster (RAC) - is in theory capable of meeting the challenges of the cloud. However, RAC is seen as being too proprietary, expensive and high maintenance by most of those trying to establish computing clouds.

For those seeking to create public computing clouds (such as Amazon) or those trying to establish massively parallel, redundant and economical data-driven applications (such as Google), relational databases became untenable. These vendors needed a way of managing data that was almost infinitely scalable, inherently reliable and cost-effective.

Google's BigTable solution was to develop a relatively simple storage management system that could provide fast access to petabytes of data redundantly distributed across thousands of machines. Physically, BigTable resembles a B-tree index-organized table in which branch and leaf nodes are distributed across multiple machines. Like a B-tree, nodes "split" as they grow and, since nodes are distributed, it can scale across large numbers of machines. Amazon's SimpleDB is conceptually similar to BigTable and forms a key part of the Amazon Web Services cloud computing environment. Microsoft's SQL Server Data Services provides a similar capability. For applications already using the ORM-based frameworks, these cloud databases can easily provide core data management functionality with compelling scalability and economic advantages - the signature of a disruptive technology.

Cloud Database Drawbacks

Cloud databases still have significant technical drawbacks, including:

- Transactional support and referential integrity. Applications using cloud databases are largely responsible for maintaining the integrity of transactions and relationships between "tables."
- Complex data accesses. Cloud databases, excel at single-row transactions. Most applications use joins and other operations.
- Business intelligence. Application data has value not only to power applications, but as information for business intelligence. Businesses will not return willingly to the pre-relational database days when business data was locked in impenetrable application data stores.

Cloud databases could displace the relational database for a significant segment of next-generation applications. An architecture that delivers the scalability and other advantages of cloud databases without sacrificing information management would be appealing. In the next part of this article, we'll look at a proposal that seems to deliver just that.

Guy Harrison is chief architect for database solutions at Quest Software.