Computing Canada

Openness Complicates Database Management

by Craig S. Mullins

The IT organization has evolved from one of centralized control to an open, heterogeneous environment. This fundamental shift has caused a rapid increase in the complexity of managing systems in general and databases, more specifically. This major shift has been accompanied by multiple related changes in the ways that DBMS products function and are maintained. The sum of these complex changes creates what amounts to an intergalactic set of responsibilities required of DBAs as they manage these new "universal" database servers.

Managing in a Heterogeneous Environment

Heterogeneity is the biggest challenge faced by database administration groups in the late 1990s. Most organizations have more than one DBMS and some have as many as ten. It is not unusual for medium to larger shops to have DB2, Sybase, Oracle, Informix, and Microsoft SQL Server installed and housing production data. The fact that each of these are relational DBMSs does little to diminish the pain of administering and managing the data stored therein.

Why is this so? First and foremost, each RDBMS vendor has implemented different procedures and syntax. Although ANSI is working on a standard SQL, none of the vendors implement plain vanilla ANSI SQL. Product-specific extensions abound. For example, it is difficult to translate Sybase's Transact-SQL into COBOL for DB2 stored procedures, or into Oracle PL/SQL. Additionally, the user interface varies from product to product making it difficult for the DBA to switch from, say, Oracle to DB2.

But the problem is even deeper than multiple relational products. Most organizations have at least one pre-relational DBMS such as IDMS or IMS that still houses important production data and must be maintained. Furthermore, heterogeneity poses problems that would exist even if all of the DBMSs were 100% interoperable. How do you implement changes on multiple platforms and keep the data in multiple DBMSs synchronized? Unless a database administration tool is used that understands the open enterprise environment of the organization, synchronization must be done manually. Furthermore application changes must be coordinated with database changes. To reduce errors and enhance integrity it the database management tool should be integrated with an automated software distribution tool. This becomes increasingly complex in client/server environments where application programs needs to be distributed to hundreds or thousands of client nodes across the network.

And to take the previous analogy a step further, shouldn't all of our management tools be integrated so that management information is shared, the user interface is common across tools, and management is automated and proactive instead of reactive? Consider the following scenario. Your database performance monitor hits a threshold indicating a performance problem. The problem requires a database to be reorganized. A job is automatically scheduled for the best time using your database reorg tool and your job scheduler (on Windows NT, Unix, MVS, or whatever). A trouble ticket is automatically sent to the help desk indicating that performance could be troublesome until the reorg occurs. The reorg runs, the scheduler automatically blocks all other jobs using that database, so the problem is resolved and the trouble ticket is removed. And all of this occurs automatically and proactively with no administrator involvement. Wouldn't that be nice?

Packaged Applications

With technology changing at such a rapid pace time is not always available to develop applications internally. As such, third party applications are increasing in popularity. Just look at the rapid growth of SAP into one of the top five independent software vendors. And SAP is not alone, other popular application vendors include Peoplesoft and Baan. Five years ago almost no one had even heard of them. Organizations today are wrestling with managing the database structures that ship with these applications. The databases must be generic so that any of the major RDBMS products can be used with the application. However, this means that the design is not optimized for the product. However, code changes can not be made to the product so organizations are struggling with how to optimize these applications and what can be changed (indexing, database parameters, cache specifications, etc.)

Data Growth and Data Warehousing

Perhaps the hottest trend in database management today is data warehousing. Data warehousing separates production transactions from ad hoc, analytical queries. Data warehousing imposes a development methodology involving data refining (scrubbing and transformation), data movement (replication and propagation), and data documentation (storing metadata in a corporate repository). The true value of data warehousing is not only providing businesses with more information, but more accurate information in a more timely manner. In this way the data warehouse enables businesses to quickly react to rapidly changing business conditions and thereby gain a competitive advantage. So the data warehouse becomes as critical in terms of administration and management as the production systems are. And, guess what, the data warehouse is often implemented in incremental data marts using multiple database structures on multiple platforms. This further complicates the job of the DBA by increasing heterogeneity.

An offshoot of the data warehousing trend is data mining. Data mining is the process of heuristically discovering heretofore unknown information and patterns lurking within your organization's data. Just imagine what happens to data access and database performance when automated data mining tasks are executed against legacy data.

All of this warehousing and mining is at the core of another key trend: an ever-increasing amount of data being stored, managed, and manipulated. Multi-terabyte databases are common and petabyte databases will be just as common by the turn of the century. And what is next—exabytes, zettabytes, and even yottabytes (see chart). We are indeed moving from VLDBs to VHDBs (Very Huge Data Bases). The net result of huge databases is the increasing difficulty of database administration and management. How do you back up a terabyte of data? Indeed, the Gartner Group has produced a study showing that the largest production database at any given point in time is not administrable given the technology and tools at hand.

Abbrev.	**Term**	**Amount**
KB	Kilobyte	1,024 bytes
MB	Megabyte	1,024 KB
GB	Gigabyte	1,024 MB
TB	Terabyte	1,024 GB
PB	Petabyte	1,024 TB
EB	Exabyte	1,024 PB
ZB	Zettabyte	1,024 EB
YB	Yottabyte	1,024 ZB

Databases Are Going Online

The rapid acceptance of the Internet and the World Wide Web as a source for information is speeding the necessity for company's to hook their databases into the 'net. Web browsers are fast becoming the de facto method for gathering and sharing information outside (Internet) and inside (Intranet) corporations. Now that Java can be used to write applications to be executed on the Inter- and Intranet by your favorite Web browser, many administrative quagmires will develop. How do you ensure that Java applets can access your corporate databases over a vast network? Furthermore, how do you tune a database that could potentially be accessed by every Web user in the world? And how is down-time managed? Anyone who has surfed the web knows that a site can be available one second, but not necessarily the next. These are new issues that DBAs are not accustomed to dealing with.

Extending the Database

Traditionally, DBMS products stored data and nothing else. But all of the major RDBMS products of today support procedural logic in the form of triggers, functions, and stored procedures, otherwise known as Server Code Objects (or SCOs for short). SCOs consist of application logic that is tightly coupled to the DBMS. Of course, not all of the RDBMSs support each type of SCO, nor do they implement SCOs in the same way. This greatly complicates the job of the DBA. But storing procedural logic in the database is here to stay because it enhances performance of client/server applications, eases security, and promotes reusability.

Last, but definitely not least, is the freight train known as object-orientation, or OO. The basic idea behind OO is that process should be encapsulated with data thereby creating complex objects (instead of the simple rows and columns we use with RDBMSs). Objects increase the possibility of reuse and provide structures for dealing with the complicated data that relational technology has traditionally struggled with (e.g., CAD/CAM drawings, bill-of-material hierarchies, etc.). All of the major RDBMS products have been re-vamped to support objects, creating what is being called a “universal” server. The advent of “universal” DBMS products further complicates database administration as objects will need to be modeled, implemented, maintained, and administered—just how do you reorg an object? Over the course of 1998 more shops will migrate to new releases of object/relational products such as Oracle8, Sybase Adaptive Server, and DB2 Universal Database. And the world of database administration will become even more complex as users implement objects and they become a component of the database environment.

Synopsis

The task of database administration is becoming more and more complex. Multiple environments with large amounts of complex data and logic are being accessed 24 hours a day and 7 days a week across the world. In this day and age, the role of the DBA is indeed an intergalactic one.