情 報 学 基 礎 20－1
（1990. 12. 18）

分散データベースの動向

アレン・マイナー
（日本オラクル）

分散データベースおよびマルチベンダ環境でのシステム・インテグレーションについて述べる。

Enterprise-Wide Information Systems -- The Future of Distributed Systems and Multi-Vendor Computing

Allen Miner / Oracle Corporation Japan

Abstract

Beginning in the late 1980s the foundations of a standard Software Architecture has been laid which will enable enterprises to integrate all their information into a single unified resource. By the end of the 1990s we should see widespread implementation of multi-vendor Distributed Relational Database Management Systems as the new "platform" that makes such integration possible. This paper provides an overview of current technology, user needs, and where the industry is headed as regards integration of distributed systems.

Enterprise-Wide Information Systems -- The Future of Distributed Systems and Multi-Vendor Computing

Allen Miner / Oracle Corporation Japan

Abstract

Beginning in the late 1980s the foundations of a standard Software Architecture has been laid which will enable enterprises to integrate all their information into a single unified resource. By the end of the 1990s we should see widespread implementation of multi-vendor Distributed Relational Database Management Systems as the new "platform" that makes such integration possible. This paper provides an overview of current technology, user needs, and where the industry is headed as regards integration of distributed systems.

Introduction

In the 1960s and 1970s computing tended to be quite centralized. A typical organization's information system consisted of an IBM (or Fujitsu, Hitachi, or NEC) mainframe running a complicated proprietary operating system that was used only by data processing professionals. The machines were so expensive and required such a high level of professional expertise to operate that an application backlog of 2 to 3 years was not uncommon.

The introduction and widespread use of personal computers, minicomputers, and workstations along with easy-to-use software such as spreadsheets, relational databases and 4GLs in the 1980s made it possible for end users to do more and more of their own processing. This helped to reduce the application backlog but it also led to the creation of "islands of information." Each division and each person had information that was only accessible to himself. It was in his Lotus 1-2-3 files or in his private relational database, or in files on his workstation and it was not any easier to get at the information on the corporate mainframe when he needed it to do important work.

In an effort to solve this "islands of information" problem, the 1980s also saw the widespread introduction of local-area and wide-area networks. Unfortunately, networking alone does not solve the problem. This is because the "islands of information" is caused by the combination of five incompatibilities.

1) Multiple Hardware Vendors
2) Multiple Operating Systems
3) Multiple Network Protocols
4) Multiple User Interfaces
5) Multiple Data Management Systems

To acheive a fully-integrated enterprise-wide information system all five of these system incompatibilities must be overcome in a way that is transparent to end-users, to application programmers, and ultimately to system (and network) managers. The network must appear to everyone on it as if it were a standalone local system. This can be accomplished in one of two ways.

First, an enterprise could require all of its end-users to conform to a single standard. For example, such a standard might require every computer users in the organization to use Hardware manufactured by IBM, running the UNIX operating system, OSI networking, Motif User Interface and the ORACLE RDBMS. I do not believe that such complete standardization is practically acheivable or desireable. I believe that each system has unique advantages and that the ideal information system will mix and match a variety of systems to take advantage of these unique strengths.

This leads us to the second way to acheive an enterprise-wide infromation system: integration. This will require software that takes advantage of standards (such as SQL) where they exist, that is available on all systems (this requires that the software be ported to both UNIX and proprietary operating systems), that supports all major User Interfaces including both terminals and Graphical Displays, that interfaces with all major network protocols and architectures and that has the ability to transparently access data in a variety of databases and file systems. An integrated enterprise-wide information system would allow users to choose whatever system components are best suited to their individual requirements but to freely integrate these into a single logical system.

How close are we to acheiving this objective of transparent enterprise-wide information systems, what remains to be done?

Standards

In the early 1980s there was considerable argument about what Database Management architecture was best. Arguments for relational vs . non-relational and various implementations of each were common. By the end of the 1980s, the industry was clearly converging on SQL as the standard platform whereby distributed data management systems will be acheived. At the present time there are still differences in the SQL implementations offered by each vendor but through the efforts of such standards organizations as ANSI, ISO, and JIS and industry consortium such as SQL Access Group and X/Open, we are moving little by little toward a convergence of the products. In the meantime, most of the independent vendors are shipping or have announced gateway products to other vendors relational (and in some cases non-relational) database systems.

Portability

In order to acheive a fully-integrated enterprise-wide information system, it is necessary that a common data management and application development enviroment be available on all platforms used by an organization. In the ideal case, it would be possible to develop an application on any platform, to test it on the same or any other platform and to deploy it on the same or any other platform. It would be possible to write applications just once and enter data just once but to replace the underlying hardware and operating system (and network, User-interface, and database management System) whenever an advance in basic technology was significant enough to warrent upgrading to it. Companies would be able to exploit the latest hardware and system software technology without

losing any of their investment in existing applications and data.

It is for these reasons that the Open Systems revolution has attracted such interest in the past several years. Unfortunately, UNIX alone, does not really make these goals acheiveable. That is because the most cost-effective high-quality end-user platforms are still MS-DOS and Macintosh. The most powerful and reliable server platforms are still the Proprietary architecture mainframes. Non-UNIX environments will continue to play a significant role in corporate information systems well into the next century. On the other hand, a single-vendor architecture like IBM's SAA can lock customers into that vendor making it impossible to take advantage of the attractive cost-performance of systems that has resulted from increasing competition. It is also a solution that today is only an architecture, a roadmap for the future. It is not possible to acheive the kind of 100% interchangeability of systems that is required for enterprise-wide integration with SAA because of subtle differences between the various environments involved.

Distributed Processing -- Client/Server Architecture

The first practical step towards distributed database management that began to become popular in the late 1980s is referred to as the Client/Server Architecture. It is also frequently referred to as cooperative processing or distributed processing. To understand what is meant by the Client/Server Architecture, it is useful to contrast it with two other possible database architectures. The traditional database architecture had Database Processing and Application Program Processing running on the same standalone computer system. This might have been a mini or mainframe computer with many terminals or it might have been a single Personal Computer. With non-relational (navigational) systems the data access language was too closely connected with the physical structure of the database (the database Management System and Application Program were too tightly-coupled) to make separating them into a server process and an application process practical. In a traditional environment, all the processing takes place on the host, the terminals do nothing.

On the low-end, as PCs started to be connected into lans the need to have shared databases became apparent. The early LAN DMBSs, took the opposite approach of host systems. Each client (this time PCs instead of terminals) would download from the file server a copy of both the application and the DBMS package. Every time a record of data was needed, the entire database file would be sent across the network from the file server to the client. This obviously could lead to tremendous network overhead and also made it difficult to manage the locking required in multi-user update applications and impossible to effectively guarantee the integrity of the database in case of a system failure.

In the client/server architecture, all database processing is done on a database server and all application processing on a separate client workstation. SQL statements are sent from the client to the server which processes the request and returns only the data actually required by the application. The Server also handles multi-user locking, and ensures database integrity and allows central maintenance of the system. Client/Server systems are used for primarily two reasons. The most important is

quality. Client/Server systems allow users to deploy the most modern user-interface technology on the front-end and the most advanced parallel-processing architecture servers or host computers on the back-end. The optimal system for each task can be integrated. The ideal architecture would also allow terminals to be mixed into the same environment for maximum cost-effectiveness based on each individual end-user's needs. The second benefit of cooperative processing is increased performance. By moving the application processing off the host, more CPU and I/O resources of the host can be dedicated to managing the database. For traditional terminal-style interfaces this benefit is maringal but distributed processing is crucial to make window-based interfaces possible because of the tremendous processing overhead involved in supporting the graphical user interface.

Distributed Processing was introduced by Oracle and Ingres in 1986 and has become a fairly common capability of relational database products by the end of the 1980s although important differences in implementation exist.

Distributed Processing in the DBMS Server

The next step after distributing application and Database processing across multiple platforms is to distribute database processing itself. Today, some database vendors support tightly-coupled symmetric multi-processor systems in such a way that one or more database server can run on each CPU in the system and by utilizing shared memory they can manage a shared database file. Today, only Oracle has announced a product capable of doing the same in a loosely-coupled parallel-processing environment. This will enable multiple machines in a VAX-cluster environment or thousands of CPUs in a massively parallel architecture (such as NCUBE, Parasys, or Meiko) to process one or more databases in parallel. Transaction rates in the 1000s of transactions per second (5-10 times the highest rates acheived on tightly-coupled SMP systems) are expected to be reached in this type of environment.

Distributed Database

The ability to spread database files and tables across multiple database servers on a network, and to do so transparently, is what we refer to as distributed database. It should be possible to join two tables that physically reside on different computers in a single query. It should be possible to update records that reside on two or more nodes in the network in a single atomic transaction. All the data in the network should appear to end-users and, even more importantly in terms of system maintainability, to application programs as if it resided on a single machine in a single database. It should be possible to build an application in a non-distributed environment and switch to a distributed database architecture without rewriting a single line of code.

Since 1986, Ingres and Oracle have supported the ability to perform distributed queries. The statement SELECT a,b FROM A, B will produce identical results whether tables A and B are on one computer or two and because the reference to the data does not include any information specific to the network (it is transparent), the network architecture can change without changing the application.

Efforts in the area of distributed database optimization and network fault-tolerance have been underway at both companies for the past 5 years.

We have reached the stage where some vendors of RDBMS packages have introduced support for programmed two-phase commit. This makes it possible to distribute and update transaction across multiple server nodes. Unfortunately, it requires application programmers to code differently for distributed and non-distributed environments. It also usually requires the programmer to explicitly specify the node on which data resides. This makes it impossible to re-architect the network without recoding applications and also introduceds the risk of programmer errors in the coordination of transactions. The ideal approach would use the identical syntax "COMMIT" to commit any transaction whether it was distributed or not. This requires substantially more development in the RDBMS engine itself and may require coordination with a transaction processing monitor such as CICS or AT&T System/T.

All distributed database products support horizontal fragmentation of tables under the control of an application but none can do it transparently. It is quite common for end-users to want to be able to manage local subsets of a given database table locally but have consolidated complete database shared centrally for the entire enterprise. This can be accomplished under application control with today's technology, or through automatic table replication if the user is willing to maintain a complete copy of all the data at each node in the network, but considerable work remains to be done in order to manage horizontal fragmentation of data subsets with a unified, consolidated base.

Conclusion

As the techonology required for transparent distributed processing and distributed database become available from more database vendors, as they are ported to more computer systems, as interface and network standards evolve and products conform to them, we are makeing steady progress toward the time when users will be able to choose any hardware, any operating system, any network protocol, any user interface, and database management system and have them all work together automatically, transparently.