THE NEW

USING MIDDLEWARE, CUSTOMERS CAN DEPLOY COST-EFFECTIVE AND HIGHLY FUNCTIONAL CLIENT/SERVER APPLICATIONS — ONCE THEY WORK OUT THE KINKS.

BY RICH FINKELSTEIN

Broadly speaking, middleware ties application components together. This definition encompasses too many products to cover in one article, so instead I'll look at significant products that can be used to manage cross-platform connectivity in a database environment. Within this context, middleware products provide several important functions, including:

- data access to heterogeneous databases;
- gateways to remote mainframe databases;
- application partitioning across multiple hardware platforms; and
- distributed updates across a homogeneous or heterogeneous set of relational databases.

Many middleware products have appeared over the last several years that have attempted to solve the various problems associated with accessing and updating databases over local and wide area networks (LANs and WANs). For the most part, progress in deploying production applications that utilize middleware has been slow — much slower than what you might first think. Customer references for middleware products are still few and far between. Most applications that use middleware seem to have a relatively simple application profile; that is, low volumes of read-only transactions. However, in some cases, middleware has been used successfully to support very large, mission-critical systems. Developing these types of systems usually takes a substantial amount of time, resources, and commitment; in other words, you must have a high tolerance for bugs and performance problems that have to be worked out of the middleware software methodically.

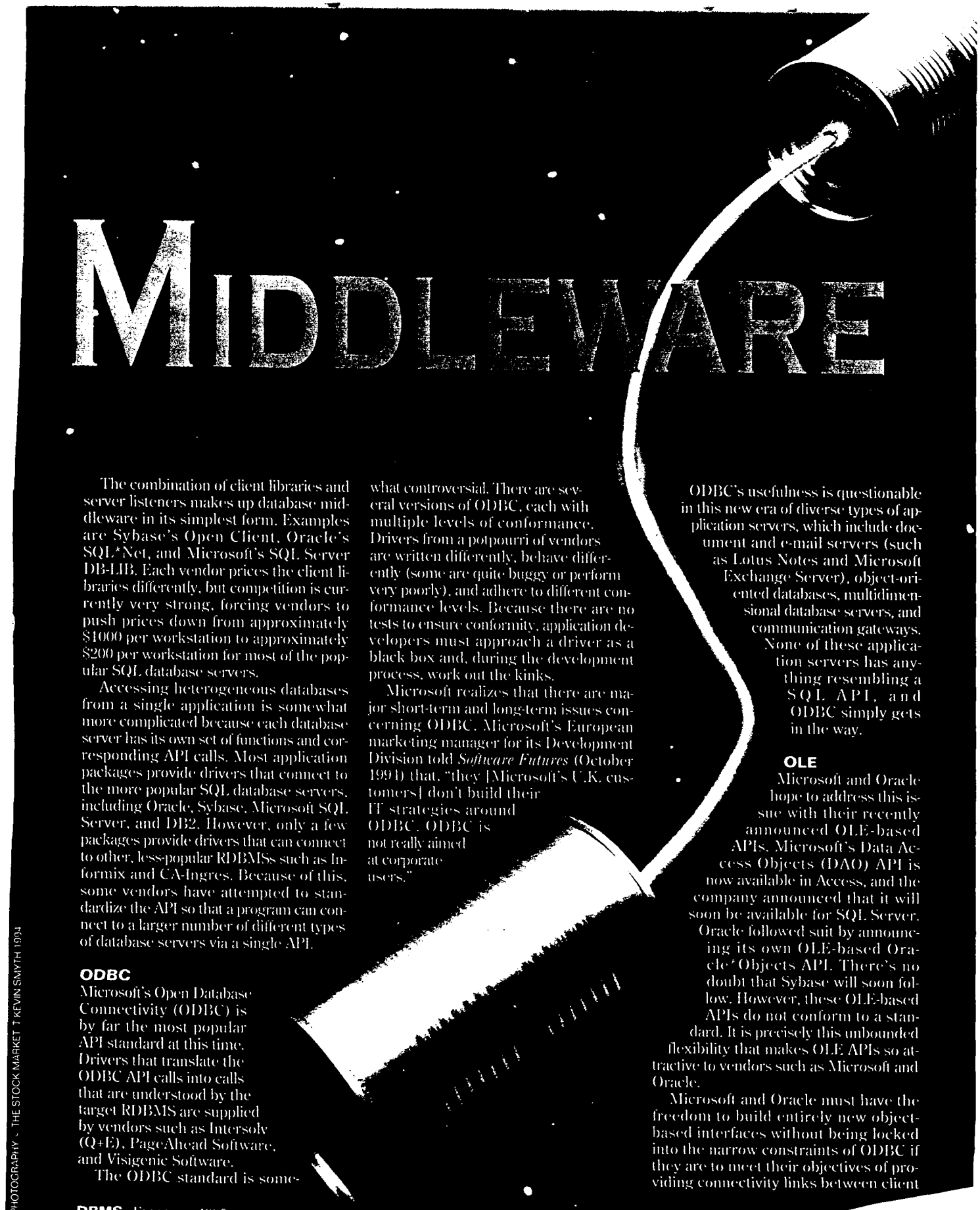
Products that were introduced several years ago to address the simpler problems of middleware, such as simple data extraction or update of mainframe-based relational databases, have reached a level of performance and stability that makes them good choices for implementing a wide variety of client/server applications. Transaction monitors and certain remote procedure call (RPC) mechanisms have also proved to be very useful for a broad set of applications. Thus, whereas middleware must still be given time to mature, there are many cost-effective products available today that work well.

Connecting to Heterogeneous Data

All SQL-based database servers come with client program libraries that include an application programming interface (API), and a set of network libraries that let client application programs attach to SQL database servers using various network protocols. On the server side of the network, a companion program called the network listener waits for and grabs messages from the client application and hands them off to the database server. The listener then routes data extraction result sets or other messages back to the client.

.....

Rich Finkelstein is president of Performance Computing Inc., a database technology consulting company based in Chicago.



MIDDLEWARE

The combination of client libraries and server listeners makes up database middleware in its simplest form. Examples are Sybase's Open Client, Oracle's SQL*Net, and Microsoft's SQL Server DB-LIB. Each vendor prices the client libraries differently, but competition is currently very strong, forcing vendors to push prices down from approximately $1000 per workstation to approximately $200 per workstation for most of the popular SQL database servers.

Accessing heterogeneous databases from a single application is somewhat more complicated because each database server has its own set of functions and corresponding API calls. Most application packages provide drivers that connect to the more popular SQL database servers, including Oracle, Sybase, Microsoft SQL Server, and DB2. However, only a few packages provide drivers that can connect to other, less-popular RDBMSs such as Informix and CA-Ingres. Because of this, some vendors have attempted to standardize the API so that a program can connect to a larger number of different types of database servers via a single API.

ODBC

Microsoft's Open Database Connectivity (ODBC) is by far the most popular API standard at this time. Drivers that translate the ODBC API calls into calls that are understood by the target RDBMS are supplied by vendors such as Intersolv (Q+E), PageAhead Software, and Visigenic Software.

The ODBC standard is somewhat controversial. There are several versions of ODBC, each with multiple levels of conformance. Drivers from a potpourri of vendors are written differently, behave differently (some are quite buggy or perform very poorly), and adhere to different conformance levels. Because there are no tests to ensure conformity, application developers must approach a driver as a black box and, during the development process, work out the kinks.

Microsoft realizes that there are major short-term and long-term issues concerning ODBC. Microsoft's European marketing manager for its Development Division told *Software Futures* (October 1994) that, "they [Microsoft's U.K. customers] don't build their IT strategies around ODBC. ODBC is not really aimed at corporate users."

ODBC's usefulness is questionable in this new era of diverse types of application servers, which include document and e-mail servers (such as Lotus Notes and Microsoft Exchange Server), object-oriented databases, multidimensional database servers, and communication gateways. None of these application servers has anything resembling a SQL API, and ODBC simply gets in the way.

OLE

Microsoft and Oracle hope to address this issue with their recently announced OLE-based APIs. Microsoft's Data Access Objects (DAO) API is now available in Access, and the company announced that it will soon be available for SQL Server. Oracle followed suit by announcing its own OLE-based Oracle*Objects API. There's no doubt that Sybase will soon follow. However, these OLE-based APIs do not conform to a standard. It is precisely this unbounded flexibility that makes OLE APIs so attractive to vendors such as Microsoft and Oracle.

Microsoft and Oracle must have the freedom to build entirely new object-based interfaces without being locked into the narrow constraints of ODBC if they are to meet their objectives of providing connectivity links between client

PHOTOGRAPHY © THE STOCK MARKET © KEVIN SMYTH 1994

applications and a wide breadth of application servers. Already, organizations are hammering vendors for links to Lotus Notes, imaging databases, and text-processing databases. OLE can potentially provide this mechanism. For now, OLE appears to be not only Microsoft's direction within its own product line, but also the direction of most of the client/server industry.

Gateways to Mainframe Databases

It is no surprise that the most mature middleware products on the market are designed to connect workstation applications to critical mainframe databases such as DB2. Despite the excitement surrounding client/server and PC computing, most corporate data still resides on mainframes. There are several reasons for the slow migration off mainframes. Mainframes are still more reliable and have more mature and well-defined data integrity and security mechanisms. An enormous amount of investment money and knowledge is associated with mainframe applications, and it takes time and substantial resources to understand the nature of these mainframe applications, much less reengineer them for minicomputers and PCs. Mainframe applications are tightly linked among themselves and to shared databases, making it difficult to migrate one application without affecting many others.

Database gateways, such as those provided by Micro Decisionware Inc.'s (MDI) Database Gateway (recently acquired by Sybase) and Information Builder Inc.'s (IBI) Enterprise Data Access (EDA)/SQL, are designed to promote gradual migration from traditional dumb-terminal/mainframe database environments into a client/server architecture. While many applications are still more than adequately serviced by traditional architectures, new business requirements often mandate the introduction of PC workstations. Using gateways, an organization has the option of keeping the database server exclusively on the mainframe platform, or using a combination of Intel, Unix, and mainframe databases to process an application. Each database can contain a logical partition of an entire database, or it can contain data that has been replicated from the mainframe database. Moving data to Intel- or Unix-based platforms tends to improve accessibility and performance, but also complicates application development and administration.

Database gateways sit between the client application and the target mainframe database. Their primary functions are to: translate LAN protocol packets into WAN packets (for example, IPX/SPX into APPC); translate client-generated SQL commands into commands that are understood by the target database; and extract data from the target database and return it to the requesting application.

Based upon most surveys, MDI Database Gateway is the most popular gateway of this type. You can install the network and SQL conversion functions as OS/2 or Unix servers, or on individual OS/2 workstations. A server-based database gateway acts as a concentrator, which provides gateway services for several workstations in a workgroup. Server solutions are more cost-effective and offload communications work from the workstation.

IBI's EDA/SQL has an advantage over MDI Database Gateway in that you can install it on Windows workstations, Unix servers, or on the mainframe. Organizations often deploy IBI's gateway on the mainframe to eliminate the need for an intermediary Unix or OS/2 server.

The mainframe component of MDI Database Gateway includes a DB2 access module that can extract data directly from a DB2 database. There are also facilities called remote stored procedures, which consist of user-written CICS programs that access non-DB2 databases such as IMS, CA-IDMS, and VSAM. Remote stored procedures let customers customize and tune access to nonrelational data sources, and are powerful mechanisms for attaching new client/server applications to legacy databases. Sybase/MDI also licenses a set of gateway modules from Trinzic Corp., called InfoHub, which can access a number of different mainframe data sources, including IMS and VSAM files. However, InfoHub has a relatively small market presence and it is difficult to judge the performance and stability levels of these modules.

IBI's EDA/SQL also offers gateway modules that access many different types of data sources. EDA/SQL has a longer history when it comes to accessing nonrelational databases — especially IBM's mainstay IMS databases. Because of this, EDA/SQL is often chosen for IMS gateway access, but performance tends to become a problem as queries grow in complexity and volume. IBI likes to advertise its ability to join any amount of data from any type of data source, but you should take this claim with a grain of salt. Unless the queries are trivial, performance will be so poor that this approach simply becomes impractical. As with MDI Database Gateway, EDA/SQL customers have the option to build their own CICS database access programs that are functionally similar to MDI's remote stored procedures.

MDI Database Gateway is usually chosen over EDA/SQL because of performance, its ability to service a larger number of concurrent users, and its bulk-transfer capabilities. EDA/SQL tends to be the choice when transaction volume is light and customers must quickly implement a mainframe gateway to a nonrelational DBMS such as IMS, but they do not have the time to write a more customized and efficient CICS program. In addition, some customers may choose EDA/SQL because it does not require an intermediate Unix or OS/2 server, thereby simplifying the environment and allowing quick deployment of a prototype or simple production application.

Don't Forget about DRDA

IBM's Distributed Relational Database Architecture (DRDA) offers an interesting alternative to database gateways. The DRDA specification standardizes the network protocol formats (APPC and TCP/IP), SQL syntax, and data representation format used to link IBM's relational databases. This allows IBM's DB2 databases on OS/2, RS/6000, AS/400, and MVS to share data without the need for intermediary gateways. DRDA tightly integrates the various DB2 platforms and provides a high degree of reliability and performance. Many companies have implemented critical online transaction-processing applications using DRDA. Usually, these applications are some mix of OS/2 workstations with DB2/OS2 and DB2/MVS mainframe databases.

Adding Value to Database Gateways

Several middleware products work in conjunction with and add value to the MDI and EDA/SQL gateways. One of the better-known products is Trinzic Corp.'s InfoPump. InfoPump is a data access management tool that lets organizations schedule periodic data extracts from a variety of data sources, and then direct the output of these requests to a local database server such as Oracle or Sybase. InfoPump includes a scripting language that developers use to specify the data sources, data targets, and data extract schedule. It connects to a mainframe database via EDA/SQL or MDI Database Gateway, or to local database servers via direct drivers. Unlike MDI Database Gateway and EDA/SQL, InfoPump is still relatively new in the market, and it is difficult to gauge its performance capacity. Customer reports vary, so you might benefit from customer checks or in-house benchmarks before signing on to the product.

InterViso, from Data Integration Inc., is attempting to go far beyond the data-integration capabilities of most middleware products on the market. InterViso's

objective is to provide complete, transparent access to a set of heterogeneous databases — what is sometimes referred to as a "federated database." To accomplish this objective, InterViso translates data types, provides common names to similar fields in different databases, supports data encoding, manages data that is replicated among several databases, provides a single view of records that are fragmented across multiple databases, optimizes distributed (multidatabase) queries, and permits heterogeneous updates without sacrificing security, transaction-management support, or the local autonomy of each database.

The key to InterViso is its Data Dictionary/Directory (DD/D), which stores a complete description of all databases in the environment. DBAs are then free to define federated data "views," which can include data from one or more heterogeneous databases. InterViso's "controller" uses the DD/D to map views to underlying physical tables and then optimizes any queries that span more than one database. The controller also controls multisite updates using a specially designed COMMIT protocol, which allows transactions to continue even if a node is unavailable. The Database Management System Interface (DBMSI) converts fields with similar meaning but different data types into fields with a common data type. InterViso can handle even very difficult situations where one field may have a numeric data type while another field with a similar meaning has a character data type. InterViso will convert the numeric data type to a character field so that the values stored in the two fields can be merged together.

Data Integration Inc. claims that InterViso can support fragmented tables (tables that are horizontally or vertically partitioned) and replicated data automatically. Considering the complexity of the problem, you might want to approach this claim with some skepticism. However, if the product works as claimed, it can be tremendously useful when you're implementing decision-support databases or migrating applications from mainframe databases. I would recommend that any customer considering InterViso should spend some time interviewing customer references and prototyping with the product in order to understand the functional and performance limitations of InterViso's distributed query and update capabilities. Even if it does not have all the capabilities you require, it may be adequate for solving many types of distributed application problems.

Partitioning Application Logic

Application partitioning — splitting application logic across multiple platforms — is now in vogue. However, application partitioning is not new to client/server processing. Stored procedures, which Sybase promoted heavily when it first introduced its SQL Server database, are good examples of simple application partitioning.

There are times when application logic must be deployed on the server, and either the logic is too complicated for stored procedures or the server does not offer stored procedures. One of the benefits of stored procedures and application partitioning is that they help minimize network traffic by moving the database access application logic onto the server. Application partitioning also offloads work from workstations, which are often overtaxed. One of the problems with client/server processing is that workstations are often incapable of supporting complex application processing. One way to get more life out of workstations is to move processing logic to the server. It is far more economical to upgrade one server (if necessary) than to upgrade hundreds or thousands of workstations.

■ You should not undertake the task of application partitioning without some forethought. Partitioned applications are more difficult to design and maintain.

You should not undertake the task of application partitioning without some forethought. Partitioned applications are more difficult to design and maintain. An application that is split up into several pieces is more complex and has more potential points of failure. You must design partitioning into the program at its inception so that logical points exist from which to divide the application. This is exactly the situation that exists today when developers design and program applications that implement stored procedures: Overly partitioned applications can result in unnecessary network traffic and unstable systems.

RPC Solutions

Products that support RPCs offer a straightforward way to partition application logic without programming complex network communication protocols. This category of products includes companies such as NobleNet Inc., Netwise Inc., and NetWeave Corp. (Wilmington, Del.). As with most products in the middleware category, RPC products have enjoyed only limited use until recently, although demand is growing as client/server applications scale up in complexity. There is typically very little reason to implement RPCs in simple workgroup environments (which represent the lion's share of the client/server applications in production at this time).

A good example of the use of RPCs is an application that my company recently helped develop. This application consisted of a local FoxPro database that needed to connect to a remote Informix database whenever a customer wanted to retrieve data that was unavailable in the FoxPro database. Because FoxPro was running under DOS in this case, we wrote a customized network protocol that sent a FoxPro-initiated SQL request over to the server. On the server, an extract program received the request, extracted the data from Informix, and sent the data back to the workstation.

Under DOS, we wrote the network communication layer that sent the requests and returned the data packets. But under Windows, you can take advantage of a product such as NobleNet, which analyzes the client and server application programs and automatically generates the client and server networking communication programs (called "stubs"). This saves a considerable amount of time, and does not require specialized network communications expertise. Most developers who can write a C program can use NobleNet to partition an application across a network.

Netwise recently introduced TransAccess Application/Integrator Workstation (A/IW), which lets developers define the communications interface using a GUI specification tool. GUI definition is presumably easier to use and maintain than generated C language code.

Several vendors have introduced application development tools that they claim can partition applications across multiple heterogeneous sites easily and dynamically. Foremost among this class of products are Dynasty and Forté. Unfortunately, the vendors are so far unable to provide evidence that their products can really perform these tasks efficiently and reliably. Because the products are so new, most customers seem to be making little or no use of application partitioning within their deployed production applications. These products may prove to be worthwhile, but I hesitate to recommend them because of the lack of proof of concept.

Transaction Managers in a Distributed Environment

As client/server applications grow in size, importance, and complexity, there is a

corresponding need for greater reliability, security, availability, and performance. On mainframes, transaction managers (TMs) — also called teleprocessing monitors (TPs) — such as CICS and IMS/DC (or IMS/TM) have always held a position of great importance. TMs help ensure predictable response time, guarantee that transactions are completed successfully, allow messaging among applications, and support a very granular level of security. Security officers using TMs can define which users can access which transactions. It is also possible to define which terminals can be used to execute specific transactions.

TMs on Unix and Intel platforms have not achieved as strong an endorsement in client/server environments as in mainframe environments, but this should change over the next several years as client/server matures and is better able to support high-volume distributed applications in a cost-effective manner.

Novell's Tuxedo and IBM's CICS are two of the most popular TMs on the market today. Tuxedo supports several Unix platforms, but the most popular are Sun, Hewlett-Packard, and RS/6000. CICS runs on OS/2, RS/6000, and the AS/400, as well as on IBM mainframes. Currently, TMs do not support Windows NT or Novell NetWare. Novell has stated that it intends to port Tuxedo to Intel platforms, but it may take some time. NetWare is simply incapable of supporting a TM reliably, and it is unlikely, for competitive reasons, that Novell is going to be in much of a hurry to support Windows NT.

TMs, like other middleware products, receive requests from an application program and route the requests to a server application, which updates the target database. TMs provide specific capabilities for managing large distributed applications. Tuxedo guarantees that the application request is delivered to the target database server platform by utilizing disk-based message queues that are not lost in case of system failure. Tuxedo also provides built-in logging to recover transactions that were in the process of updating but were not completed at the time of system failure. It can also support heterogeneous database updates with a two-phase commit protocol, although I have not seen this feature in any production application at this time.

Up until now, writing Tuxedo code has been cumbersome. However, newer application development tools such as JYACC's JAM 6 have high-level language support for Tuxedo that is much easier to write and maintain. JAM 6 automatically generates the client and server stubs that interface with Tuxedo and the target database server. A word of caution: JAM 6 is a new product and has not been widely deployed, but the architecture merits a look.

IBM's CICS has enjoyed widespread popularity, primarily because it provides reliable connectivity across IBM's hardware and database platforms. Organizations commonly use CICS as the means of linking OS/2-based applications to mainframe DB2 and IMS databases. In my own consulting practices, I have noticed that when large institutions need to deploy critical applications that access mainframe data, they usually turn to CICS and OS/2. A large number of developers are familiar with the CICS API because of its long-term use on the mainframe. This makes CICS more accessible to large organizations that have developed a substantial CICS knowledgebase over the last 20 years.

Both Tuxedo and CICS offer viable solutions for companies that demand system integrity and cannot tolerate the large integrity holes that exist in current client/server software and hardware products. My primary concern with Tuxedo is the fact that it is now owned by Novell. In the past, Novell has shown little interest in Tuxedo because of its own obvious preoccupation with NetWare. UnixWare has also suffered from neglect. In recent months, however, Novell has shown renewed interest in promoting Tuxedo, possibly because of its long-term value to Novell's enterprise connectivity strategy.

IBM's CICS is firmly entrenched in the market, and it is a good choice for IBM platforms. Until recently, IBM and Hewlett-Packard were cooperating in using the Encina TM as the core technology for both IBM's RS/6000 CICS system and Hewlett-Packard's forthcoming TM. However, when IBM purchased Encina, Hewlett-Packard backed out of the Encina agreement, and the future of the CICS and Encina non-IBM operating platforms is unclear.

Another concern with CICS for RS/6000 is that it implements a layered architecture that consists of Encina technology as well as other connectivity layers. This layering tends to impact performance, and some customers report that the RS/6000 version of CICS is just too slow. IBM may decide to merge all of the code or use the OS/2 CICS codebase for future releases of CICS on IBM and Unix platforms. Whatever happens concerning CICS on Unix, CICS will probably continue to do well because of its maturity and proven capabilities.

The Future of Middleware

Two years ago, I wrote an article for *DBMS* on middleware ("Making Connections Across the Enterprise," January 1993) and there is certainly a lot more to talk about today than there was then. Interestingly enough, while products with great-looking features continue to pour into the market, the adoption of middleware has been rather slow. I believe that most organizations are still getting their feet wet with client/server and they are still trying to establish a firmly grounded knowledgebase, infrastructure, and migration plan before moving ahead with client/server. During this waiting period, client/server technology — including middleware technology — will have a chance to mature.

The best strategy at this time is to rely mainly on proven middleware products and gradually add more technology as in-house knowledge grows. Several solid products, such as MDI Database Gateway, EDA/SQL, CICS, and Tuxedo, have been around for some time and have had a chance to find comfortable positions in distributed client/server applications. Other products, such as InfoPump and RPC programs, have also enjoyed a good amount of success. As middleware gradually becomes more mainstream, customers will be able to deploy cost-effective and highly functional client/server applications. ■

- Data Integration Inc., 11965 Venice Blvd., Ste. 305, Los Angeles, CA 90066; 310-313-9150 or fax 310-313-9151.
- Dynasty Technologies Inc., 500 Technology Dr., Ste. 100, Naperville, IL 60563; 708-355-8300 or fax 708-355-9345.
- Forté Software Inc., 1800 Harrison St., 15th Fl., Oakland, CA 94612; 510-869-3400 or fax 510-834-1508.
- IBM Corp. Old Orchard Rd., Armonk, NY 10504; 800-342-6672 or 914-765-1900.
- Information Builders Inc., 1250 Broadway, 30th Fl., New York, NY 10001-3782; 800-969-4636, 212-736-4433, or fax 212-268-7470.
- Intersolv Inc. (Q+E), 3200 Tower Oaks Blvd., Rockville, MD 20852; 800-547-4000 or fax 301-984-3047.
- JYACC Inc., 116 John St., 20th Fl., New York, NY 10038; 800-458-3313. 212-267-7722, or fax 212-608-6753.
- Microsoft Corp., One Microsoft Way, Redmond, WA 98052; 800-426-9400. 206-882-8080, or fax 206-883-8101.
- Netwise Inc., 2477 55th St., Boulder, CO 80301; 800-733-7722, 303-442-8280, or fax 303-442-3798.
- NobleNet Inc., 337 Turnpike Rd., Southboro Technology Park, Southboro, MA 01772; 508-460-8222 or fax 508-460-3456.
- Novell Inc., 122 East 1700 South, Provo, UT 84606; 800-453-1267, 801-429-7000, or fax 801-429-5155.
- Oracle Corp., 500 Oracle Parkway, Redwood Shores, CA 94065; 800-633-0596, 415-506-7000, or fax 415-506-7200.
- PageAhead Software Corp., 2125 Western Ave., Ste. 301, Seattle, WA 98121; 800-967-9671, 206-441-0340, or fax 206-441-9876.
- Sybase Inc., 6475 Christie Ave., Emeryville, CA 94608; 800-879-2273, 510-922-3500, or fax 510-658-9441.
- Trinzic Corp., 101 University Ave., Palo Alto, CA 94301; 800-845-2466, 415-328-9595, or fax 415-321-7728.
- Visigenic Software, 951 Mariners Island Blvd., Ste. 460, San Mateo, CA 94404; 415-286-1900 or fax 415-286-2464.