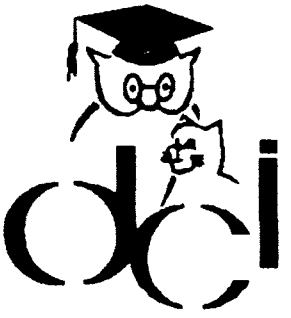


digital consulting inc.

presents

Distributed DBMS: An Evaluation

A written Evaluation for

Ashton Tate

November 15, 1990

Digital Consulting, Inc.
204 Andover St.
Andover, MA 01810 USA
508/470-3870

DISTRIBUTED DBMS: AN EVALUATION

by George Schussel

CHAPTER 1 - HISTORY AND DEVELOPMENT OF THE MARKET FOR DISTRIBUTED DBMS SOFTWARE

The market for modern distributed DBMS software started in 1987 with the announcement of INGRES-STAR, a distributed relational system from RTI of Alameda, California. Most of the original research on distributed database technology for relational systems took place at IBM Corporation in IBM's two principal California software laboratories, Almaden and Santa Theresa. The first widely discussed distributed relational experiment was a project called R-Star, developed within IBM's laboratories. It is because of IBM's early use of the word STAR in describing this technology that most vendor's distributed database systems names have incorporated "STAR" in one form or in another.

Today, the market for distributed DBMS is almost entirely based on the SQL language and extensions. (Principal exceptions being Computer Associates with its distributed DATACOM, and Fox Software with its newly announced Fox Server.)

There are three broad segment to the market

1. True distributed DBMS
2. Distributed access (remote data access)
3. Client Server

True distributed DBMS products can be considered to occupy the Mercedes Benz segment of the the market place. These products support a local DBMS at every node in the network along with local data dictionary capability. Their capability will be discussed in Chapter 2. The market for true distributed DBMS is growing slowly for two reasons: 1) users aren't sure of how to use the products and 2) the vendors are taking the better part of a decade to deliver full functionality. One important unanswered concern of companies who want use true fully distributed DBMS environments is the cost of the communications for functions that have historically been run internal to single computers.

Distributed access can properly be thought of as a subset of technologies that are being delivered by those vendors selling true distributed DBMS or client server DBMS technologies. The goal of distributed access is to provide gateways for access to data that is not local. The demand, of course, is greatest for the most popular mainframe file and database environments such as IMS, DB2, VSAM, and Rdb. Local DBMS capability is not a requirement for distributed access. Most vendors provide a piece of software known as a requestor to be run in the client side of the RDA environment. Some of the products in this market are not finished gateways but toolkits for users to build their own custom gateways.

If true distributed DBMS products are the top of the market then client server DBMS engines are the Fords and Chevrolets. By accepting some reduction in functionality from the definition of the fully distributed DBMS, a user is able to use client server technology to build a distributed computing environment which will run exceedingly well with today's hardware and networking techniques.

The market place for client server approaches is going to be far larger in dollar volume than for either true distributed DBMS or gateway technology. The leading vendors of servers, however are also likely to be the leaders in selling gateway access solutions.

As vendors improve the software capabilities of their client server systems it is likely that functional differences between client server and true distributed DBMS products will tend to disappear. I don't predict this to happen before the mid 1990s.

The functionality delivered by today's client server systems is not that different from true distributed DBMS. The key difference is that a client server approach only has a DBMS and Dictionary at certain designated nodes where the data resides. The client program is required to navigate to the correct server node by physically knowing which particular server to access for the necessary data.

The idea for client server computing grew out of database machine vendor approaches. Sybase's Robert Epstein had worked for Britton Lee when he came up with the idea of creating a database machine environment, but with a server that was a virtual machine rather than a physically unique piece of hardware. The systems software, then, was separated into a front end (client) which ran the program (which would be written in a 4GL) and a back end (server) which handled the DBMS chores. The advantage of this idea was that the back-end virtual database machine could physically be moved out onto a different piece of hardware whenever desired. What made this approach different from the Britton Lee approach was that Sybase planned for the server to be a generic VAX, UNIX, or PC machine rather than unique custom build database hardware. By moving the database machine into a standard piece of hardware Sybase picked up the advantage of vastly improving price performance in generic small systems.

At about the same time that Epstein was starting Sybase, Umang Gupta (then a Senior Oracle executive) had come up with the same idea and left Oracle to form Gupta Technologies.

Most other SQL DBMS vendors have jumped into the client server game. An exception is IBM, which while talking about "client server" really means true distributed computing. IBM is building a fully functional distributed architecture for its SQL products, DB2, SQL/DS, SQL/400, OS/2EE. IBM is taking several years to develop this approach.

Distributed DBMS is one of the most interesting areas of the large systems DBMS market today. (Large systems here are defined as 80386 on the small end to Cray at the top.) With the emergence of SQL as a standard, the principal ways that DBMS vendors are differentiating their products is by adding function in:

distributed or client server computing
support for Object approaches
addition of database semantics
adding more relational functionality (typically semantics)

No vendor can afford to ignore distributed capabilities.

It is clear that the old line DBMS companies such as Cullinet and ADR had a market shake out two to four years ago. Curt Monash of the Paine Webber Research Group was the most vocal analyst predicting the demise of the mainframe DBMS market. His predictions were made on the basis of market saturation, ascendency of DB2, and slow growth of mainframes.

Some analysts believe that the SQL server and distributed database market place is in for a comparable shake out now. The SQL DBMS vendors have entered choppy financial waters the last few months. Oracle's stock has dropped from 30 to 6 while the company has announced money losing quarters. Sybase has had a 15% layoff. Ingres is being acquired by ASK Computer Systems.

Once again, another Paine Webber analyst, Robert Therrien, is predicting a collapse in the DBMS industry. The following is quoted directly from his October, 1990 commentary on this subject.

> "The independent database market is in the early stages of its death throes; death for many vendors could come swiftly. Much as with the other software business that grew up around filling holes in hardware vendor's operating systems we believe the database engine has passed its prime. Most vendors did not change their strategies in time."
>
> "Why is a shake out occurring? Simple. The hardware vendors now supply database engines for free (DEC's Rdb, IBM's AS/400) or at low cost (IBM's DB2). For customers, this is actually a good thing. By making the database engine part of the operating system hardware vendors can put parts of the engine in microcode, speeding up the performance in some cases by an order of magnitude. With the exception of the UNIX database engine market, the product space now being filled by hardware vendors is now either closed or closing. Even in the UNIX engine business (where many database vendors are planning duck for cover) price competition is intense."

I do not have such a negative view of the market. Therrien believes that servers are "commodity" items that respond identically to a call in SQL. Theoretically, that is an interesting point of view, but in reality the differences in technical capabilities amongst different vendors are quite substantial. The independent vendors are also able to afford the advantage that made Oracle such a success - cross platform compatibility. For the most part hardware vendors choose not to provide such a capability.

CHAPTER 2 - DISTRIBUTED DBMS TECHNOLOGY

Distributed database software has to provide all of the functionality of multi-user mainframe database software but allow the data in the database itself to exist on a number of different but physically connected computers. The kinds of functionality the distributed DBMS must supply include maintenance of data integrity by automatically locking records and rolling back transactions that are only partially complete. The DBMS must attack deadlocks, automatically recovering completed transactions in the event of system failure. There should be a capability to optimize data access for wide variety of different application demands. Distributed DBMS should have specialized I/O handling and space management techniques to insure fast and stable transaction throughput. Naturally, these products must also have full database security and administration utilities.

Lead by Ingres Corp. (formally RTI) industry analysts have agreed on the definition of what functions above and beyond a single system, a distributed DBMS needs to perform. A quick discussion of these functions is listed below. It isn't useful to view this discussion as a feature checklist, since there is a great disparity between performing these functions at a minimum level and accomplishing them at an advanced level. There is a general feeling among the top industry analysts that Ingres provides the highest technical functionality here with Sybase and Interbase providing reasonable seconds. Please note that even though Ingres is the most advanced product available today it still is only about half way toward a full level of distributed functionality.

Requirements for Distributed DBMS

1.Location transparency

Programs and queries may access a single logical view of the database; this logical view may be physically distributed over a number of different sites and nodes. Queries can access distributed objects for both reading and writing without knowing the location of those objects. A change in the physical location of objects without change in the logical view requires no change of application programs. There is support for a distributed JOIN. In order to meet this requirement it is necessary for full local DBMS and data dictionary to reside on each node.

2.Performance transparency

It is essential to have a cost-based software optimizer to create the navigation for the satisfaction of queries. This software optimizer should determine the best path to the data. Performance of the software optimizer should not depend upon the original source of the query. In other words, because the query originates from point A it should not cost more to run than the same query originating from point B. Technology in this field of software optimization is rather primitive at this time and will be discussed further below.

3.Copy transparency

The DBMS should optionally support the capability of having multiple physical

copies of the same logical data. Advantages of this include superior performance from having local rather than remote access to data, and non-stop operation in the event of one site going down. If a site is down, the software must be smart enough to re-route a query to another source where data exists. The system should support "fail over reconstruction". This means that when the down site becomes live again that the software automatically reconstructs the data at that site to make it current.

4.Transaction transparency

The system supports transactions that update data at multiple sites. Those transactions behave exactly the same as others that are local. This means that transactions will commit or abort. In order to have distributed commit capabilities a technical protocol known as a 2-phase commit is required.

5.Fragmentation transparency

The distributed DBMS allows a user to cut relations into pieces horizontally or vertically and place those pieces at multiple physical sites. The software has a capability to recombine those tables into units as necessary to answer queries.

6.Schema change transparency

Changes to database object design need only to made once into the distributed data dictionary. The dictionary and DBMS automatically populate other physical catalogs.

7.Local DBMS transparency

The Distributed DBMS services are provided regardless of brand of the local DBMS. This means that support for remote data access and gateways into heterogeneous DBMS products are necessary.

Four ways to distribute data

Most vendors are taking many years to develop software that offers full distributed DBMS capability. As a way of bringing its distributed SQL products to market, IBM has proposed a phased implementation of four discrete steps enroute to distribution of data. These four principal steps are defined below.

Extracts - the ability to extract data simply means that there is a batch process which unloads and reformats operational data into a relational view. For example, IBM's DXT allows for batch unloading of IMS and reformatting into DB2. This extraction is manually managed.

Snapshots - are becoming a popular technique among many vendors. A snapshot is an extract as defined above along with a date and time stamp. The advantage of a snapshot is that after it's defined to the system, it is automatically created and managed. Snapshots are read-only and are effective for providing decision support access to true production data where operations personnel do not want live access to the production data (normally for performance reasons).

Distributed tables - Distributed tables can be thought of as the first level of true, real time, read/write distributed DBMS functionality that meets requirement 5) mentioned above. A system which can support distributed tables will normally manage a single physical copy of data to support the system's logical views.

Replicates - Replicates are a more sophisticated version of distributed DBMS capabilities mentioned under copy transparency above. This can be thought of as support for a single logical view by up to "n" physical copies (of the same data). These data replicates must be updatable (not snapshots). At a minimum, updatability of physical data replicates will require a software optimizer (as discussed below) and a 2-phase update commit protocol.

Software Optimizers

When there are different physical sites involved, the difference in cost between the best and worst ways of accomplishing a function such as a JOIN can easily be millions to one. Because of this, a distributed DBMS absolutely must have a cost-based software optimizer. Without a cost based optimizer, navigation to data must be under programmer control, violating a basic precept of relational theory (this is what must be done with Oracle). Without a cost based optimizer only known queries can be handled, since the performance of an unanticipated query may be impossible.

A reasonable software optimizer has to be intelligent enough to ask tough questions and to develop a correct search strategy based upon the answers to those questions. Examples of types of issues that should be handled are:

How busy are the various machines on the network?
What are relative speed of these machines?
What are the table sizes that have to be accessed?
What is the line speed between various nodes of the network?
How busy are those lines?
How are the tables organized?
What are the access patterns in indexes?
Where should software optimizer itself run?
etc.

Two-phase commit protocol

The goal of the 2-phase commit protocol is to allow multiple nodes to be updated in synchronized fashion as result of a single group of SQL statements which must be committed or rejected together.

The general procedure is as follows:

1. One node is designated as a master; the master sends notice of an upcoming query out to all of the slaves.
2. The slaves respond with ready messages when all of the data necessary for the protocol is available.
3. The master sends out a "prepare" message to the slaves.
4. The slaves lock the necessary data and respond with "prepared" message to the master.
5. The master sends a "commit" message to the slaves.
6. The slaves respond with a "done" message.

For the DBMS software vendor, developing a 2-phase protocol is one of the most challenging tasks in developing a distributed DBMS. Additional complexity in creating this software comes about from the fact that there are different types of

failure nodes and the software has to handle and recover from any combination of failure over all environments supported. For the user, operation in an environment requiring a 2-phase commit may be very costly. The cost comes about from the fact that use of a 2-phase commit requires an extra round-trip message over what happens in single computer system.

There are no standards for implementation of a 2-phase commit. Different vendors have come up with different partial implementations. It is likely that we will see a future ISO standard dealing with 2-phase commit protocol.

At the current time some functionality of 2-phase commit protocol is available from the following vendors:

Sybase, Emeryville, California
Ingres, Alameda, California
Interbase, Bedford, Massachusetts
DEC, Marlborough, Massachusetts
Empress, Greenbelt, Maryland
Computer Associates, Garden City, New Jersey

Problems of Distributed Database Technology

The advantages of distributed processing and distributed DBMS are well documented elsewhere and need not be repeated in this analysis. It is worth our while however, to provide a quick summary of some of the problems associated with this technology.

1. Communication costs can be quite high; using a 2-phase commit protocol generates lot of communications traffic.
2. There is need for gateway technology to handle SQL differences amongst the different DBMS vendors.
3. The predictability of total costs for distributed queries is variable. In other words it is hard to predict ahead of time how much it is going to cost you to get a job done.
4. Supporting concurrency along with deadlock protection is a very difficult technology.
5. Supporting full recovery with fail over reconstruction is very expensive.
6. Performing a JOIN across different physical nodes is very expensive using today's technology and networks.
7. Some advanced relational functions that are reasonable in a single computer are difficult and expensive across a distributed network, eg the enforcing of semantic integrity restraints.
8. The job of the database administrator is more difficult than in a single computer because all of the existing skills and requirements are still there, plus the integrity, optimizer, communication and data owner issues of the distributed world.
9. Data security issues are not well understood or proven. It would appear that a distributed environment is more susceptible to security breaks than a database contained in one box.

CHAPTER 3 - CLIENT SERVER COMPUTING

The history of Client/Server computing was discussed in chapter 1; here we will proceed with a definition of the technology and quick overview of the market.

Client/Server computing consists of three principal components:

- Client
- Server
- Network

The client is where the application program runs. Normally, the client hardware is a desktop computer such as an IBM PC. The application program itself may be written in a 4GL or in common 3rd generation languages such as C or COBOL. The screen forms run on the client. The control of the overall computing environment also comes from the client, which does not have control of its own data, but generates an SQL call.

The network is responsible for connecting client and server. Normally, the network consists of some kind of wire along with the communications card in both the client and server boxes. The communications software normally handles different types of communication standards such as LU6.2 and TCP/IP. Typical network environments provide support for multiple clients and servers.

The server is responsible for executing the SQL statement received from the client. Sometimes the data request is not pure SQL but it can be a remote procedure call which would then trigger a series of already existing SQL statements on the server. The server is responsible for optimization of the SQL, in other words, determining the best path to the data. The server manages the transactions. Some server technologies support advanced software capabilities such as stored procedures, event notifiers and triggers. The server is also responsible for data security and validation of the requestor. A server handles additional database functions such as

- Concurrency Management
- Deadlock Protection and Resolution
- Logging and Recovering
- Database Creation and Definition

The data dictionary runs on the server.

For most typical business applications the concept of database client/server computing is an outstanding fit. The server can be a powerful PC or mini computer running multi-user, multi-tasking server software. The client is a smaller but still powerful PC, which has the power of running applications.

The advantages of client/server computing are overwhelming, have been recounted elsewhere and will not be enumerated here - except to point out that client/server computing provides the industrial strength security, integrity and database capabilities of mini computer or mainframe architectures while allowing companies to build and run their applications on PC and mini computer networks. The use of this hardware and software combination can cut 90% of the costs of the hardware/software environment for building these "industrial strength" applications. I frequently recommend client/server computing as the preferred

technology for downsizing and implementing cooperative processing applications. A more in depth analysis of the advantages of client/server computing can be seen at many DCI conferences - such as the following:

Schussel & Yourdon On Emerging Software Technologies
Schussel on Application Development in the 1990s
Database World
DCI's Downsizing Conference

The first generation of PC software occurred in the 1980s. Popular products such as WORDPERFECT, 1-2-3, and dBASE caused the sale of 50 million PCs in the just completed decade. Most uses of the PCs in 1980s were for applications that were substantially different from mainframe and mini computer MIS "glass house" applications.

The 1990s will witness the second generation of PC software which will provide true integration between mainframe MIS approaches and PCs. Database client/server technologies will lead the charge in this area.

The kinds of capabilities that are available today with client/server computing are astounding. I have witnessed both Gupta and Sybase running on 386 PCs processing 8 - 12 TP1 transactions/second. PC hardware can support disks with 16ms access time and 2-3MB transfer rates. Such a machine can be configured with several 100MB of disk at a price of under $10,000. Its TP1 processing rate would be adequate to support up to 250 automated teller machines on a single server. If Ethernet is used as the communications environment, the network has a capability of handling up to 100 transactions per second. This kind of low cost PC oriented transaction processing environment can easily save companies 80% or more on cost of implementing low speed transaction processing environments.

Aside:
On the issue of whether Interbase should be ported to the OS/2 environment, a reasonable analysis would look at both technical and marketing factors. This analysis is beyond the scope of this study, but it must be pointed out that if the product is to be seriously proposed to the market where dBASE is traditionally strong, an OS/2 version is a requirement. (A discussion of this with Starkey indicated that the technical port had already been accomplished and that the real issue was the availability of tools for the OS/2 market.)

To play in the normal commercial market, Interbase will have to significantly change its approach to marketing. General industry exposure will be necessary and it certainly would be wise to play up the Ashton Tate connection.

I would be happy to follow this report with a more in depth study of the issue of porting to OS/2. That study could be completed before the end of this year. Exact pricing would depend on what specifics Ashton Tate wished in the report, but it should be in the ball park of $10,000. I would recommend an in depth look at OS/2EE as part of that study. By the time an Interbase port could be concluded, OS/2EE Data Manager will be a formidable competitor in this market.

There is no reason to believe that client/server computing needs to be relegated to the low end of the transaction processing spectrum. It is very reasonable to think of products like Oracle and Sybase in combination with high-end machine servers from companies such as Solbourne, Pyramid, Concurrent, Compaq, IBM or DEC. Using high-end server hardware with parallel 386, 486, 586 chips and/or multi-processing RISC chips and open operating systems (UNIX, OS/2 & LAN MANAGER) gives a vendor ability to build a machine with 100's of MIPS processing power and 15-20 gigabytes of data at a cost of well under $500,000. Combining this technology with SCSI and/or IPI channels allows a configuration of new technology hardware and database server to replace a $14 million System 390 at a savings of 95%.

CHAPTER 4 - REMOTE DATA ACCESS

There is a major market demand for products that can provide access into data located in diverse heterogeneous file and DBMS formats. Very few companies have only one type of file or database management system installed. With a total staff of 70, DCI has three different DBMS products running production systems. The proliferation of standards typically comes about because of the purchase of software packages with different embedded DBMS'. Companies who can create gateway paths to popular file formats will be successful in selling their products.

Gateways can be thought of as translation and connectivity devices from various tools and applications, normally on the desktop to various servers and foreign DBMS' running on remote host computers. The role of the gateway is to translate the syntax and semantics from one system to another. These translations have to be able to handle differences in

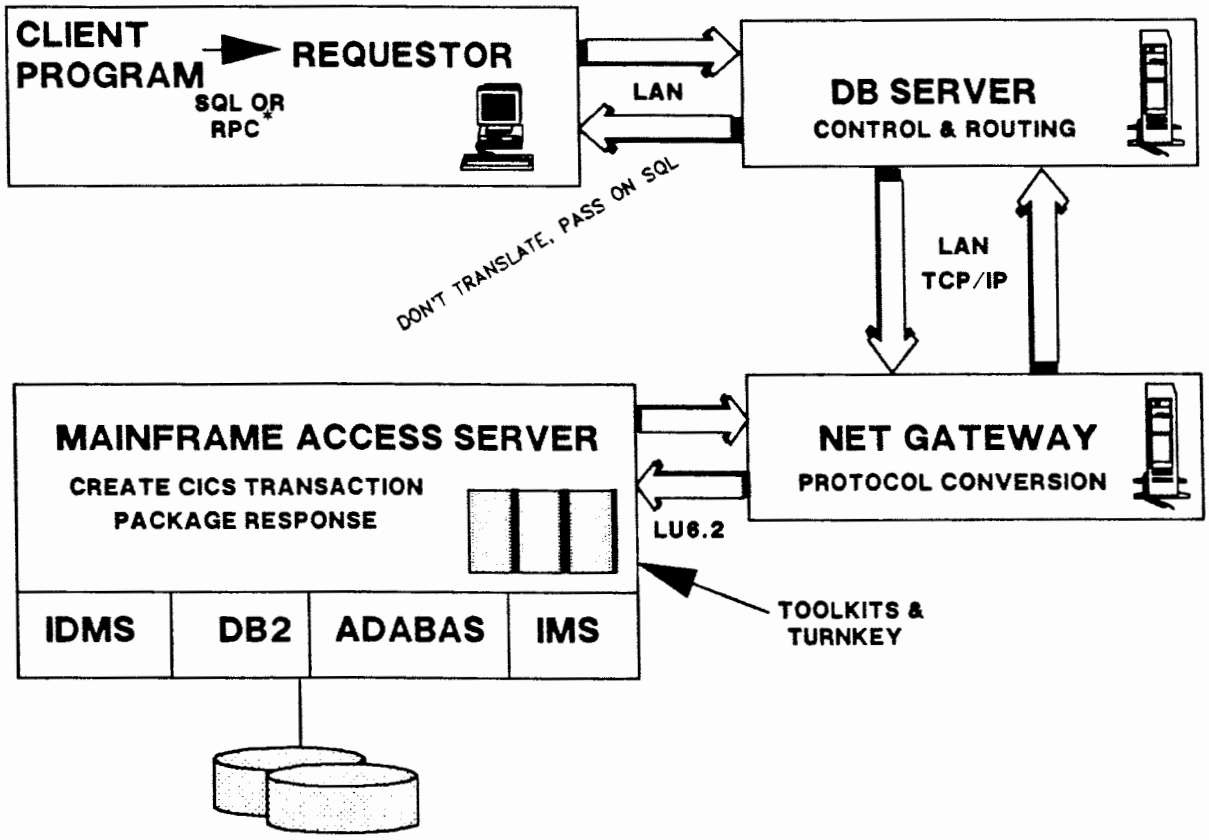
SQLs
APIs
Catalogs
Error Messages
Communication Protocols
Logging Schemes
Back-up and Recovery Schemes

Gateway and remote data access technology has normally been considered part of the distributed database product community and so it is natural to expect that the leading client/server and distributed DBMS companies would be creating products in this arena. And it is true that companies like Gupta Technologies and Sybase are among leading vendors. Information Builders of New York and Micro Decisionware of Boulder, Colorado are two companies that are not normally considered leading DBMS vendors, but are leaders in RDA technology.

Generic Gateway

The attached diagram illustrates a generic gateway technology (one that is very similar to the Sybase approach). A client program issues an SQL call or RPC to a piece of requestor software running in client machine. That call or SQL is passed on unchanged to the database server which can be a real or virtual machine running on the LAN. The database server is responsible for the control and routing of the call. In other words, it knows where to send a message to. Again, on the LAN the message is sent to a network gateway server. Like a previous environment this gateway server may be in same physical machine as a database server or it may be discrete. The gateway server is responsible for doing a protocol conversion which allows it to communicate with the mainframe. In this case our mainframe is IBM environment and an LU6.2 message is passed on to the mainframe. A PC sourced message is not likely to be something which the mainframe understands and so there needs to be software on the mainframe which takes the message from the network and converts it to a CICS transaction. That transaction then is run against the appropriate mainframe database package. Once data is extracted the procedure is reversed.

HOW TO DO A GATEWAY



** REMOTE CALLS ALLOW PRECOMPILED PROCEDURES TO RUN ON MAINFRAME*

CHAPTER 5 - SYBASE

Sybase is the company most responsible for changing the perception that relational DBMS' cannot handle transaction processing applications.

SQL Server runs under various UNIX systems, OS/2, and VAX/VMS. The OS/2 version of SQL Server requires NAMED PIPES network support, so you must install a version of LAN Manager (.e.g., 3Com 3+Open, the Ungermann-Bass version, IBM LAN Server), or Novell's NetWare Requestor for OS/2.

Database engine features

SQL Server has limited distributed update support. It comes with function calls for coordinating updates across multiple databases, but it is the programmer's responsibility to issue the correct function calls. SQL Server supports remote procedure calls that allow transactions to execute procedures on other SQL Servers.

Sybase has an Open Server product that allows developers to build gateways between Sybase and other DBMS' such as Rdb. It also has built and offers a gateway to DB2.

It also has disk mirroring and fault tolerant features. Mirroring allows an organization to keep two exact copies of a database (usually on two separate disks). If one disk fails, then Sybase will automatically use the other disk without interrupting operations. Mirroring is crucial to many OLTP applications that require fault tolerant operation. In order to engage mirroring, the database administrator issues a new DISK MIRROR command. Disk mirroring can be executed even if Sybase is in operation, so that it will not interfere with twenty-four hour processing.

SQL Server supports referential integrity and other business rules with triggers. Triggers are small SQL programs, written in SQL Server's Transact-SQL language, that are stored in the DBMS catalog. Each trigger is associated with a particular table and a SQL update function (e.g., update, delete, and insert). They are automatically executed whenever a transaction updates the table. You can write triggers to enforce any database validation rule, including referential integrity. (OS/2EE's definition of referential integrity by DDL statement stored in database tables is superior).

Since they are stored in the catalog and automatically executed, triggers promote consistent integrity constraints across all transactions. The triggers are easy to maintain because they are stored in only one place - the DBMS catalog. Rules are enforced for any application that accesses the database, such as spreadsheet programs.

SQL Server also stores rules in its catalog. Rules apply to columns, and you use them to specify user-defined data types and range checks.

SQL Server's stored procedures are similar to triggers. They are Transact-SQL programs that are stored in the DBMS catalog. Any applications (e.g., databases and spreadsheets) can call a stored procedure. Instead of executing one SQL

command at a time, stored procedures execute several commands simultaneously - without any further interaction with the application.

This saves a considerable amount of network overhead and can boost performance by 40% or more. Since Transact-SQL is a full language, developers can write complete procedures with branch and control logic, assignment operators, and error checking capabilities. Oracle's OTEX and SQLBase's chained-SQL do not have these features.

Server performance

SQL Server implements a multi-threaded, single server architecture. This type of architecture is also used by Gupta's SQLBase. Multi-threaded servers perform most of their work and scheduling without interacting with the operating system. Instead of creating user processes, multi-threaded servers create a thread for each new user. Threads are more efficient than processes, and they use less memory and CPU resources. In contrast, Oracle and XDB use a process/user architecture.

Its multi-threaded architecture enables SQL Server to efficiently service a large number of users. It can service 40 users simultaneously on a 10 MB 33MHz Compaq with only minor degradation in performance. However, SQL Server's single server architecture does not allow it to take advantage of multiple processors. However, Sybase says it is working on a "virtual server" architecture that will create multiple servers on a single machine.

Sybase uses page level locking (as compared with row level locking in Oracle). This should hinder OLTP performance, but at a practical level, doesn't seem to be an issue.

SQL Server uses a clustered index, which means that the table is kept in the same physical order and page as the key index. This improves performance by reducing head movement in database operations which frequently access data in index order, especially if you write a lot of reports in index order.

Most other DBMS' must use indexes to sequentially retrieve a range of records or a whole table. This means that the transaction must perform at least one index I/O operation and one data I/O operation for each record. Often, DBMS' must perform more. Clustering reduces the number of I/O operations by eliminating the index I/O operations and clustering data in the same database page. For many customers, this feature has made an important performance difference in the success of an OLTP application.

Cursors

A unique weakness of SQL Server is its poor support for the concept of cursors. It does not support the standard IBM SAA application "cursor" programming interface. A cursor stores the results of a SQL query and allows a program to move forward through the data one record at a time. Sometimes, as in the case of SQLBase, a programmer can move backward within a cursor. Without a cursor, it's harder to program transactions that must browse through data. It is hard to think of another SQL DBMS product that doesn't support cursors. Support for cursors will be part of

Release 5, which is expected sometime in 1991.

Tools

SQL Server's APT-Workbench toolkit has been considered weaker than many competitors and until summer '90 was not available at all under OS/2. Conversations with users of the latest versions of APT indicate much improved levels of satisfaction at this time. Other popular products that can be used to develop Sybase applications are DATAEASE, PARADOX, SQLWINDOWS and ADVANCED REVELATION.

Text and Image Data types

SQL Server Version 4.0, and later (available on UNIX platforms) supports TEXT and IMAGE data types. IMAGE data types are binary data. TEXT data types are printable character strings. IMAGE data types are binary data. Strings can be as long as two gigabytes. A table can contain up to 250 TEXT or IMAGE columns. These are defined in the CREATE TABLE statement using the TEXT or IMAGE data type keywords.

TEXT and IMAGE data types are stored as linked lists of pages. A pointer in the data row stores the value of the first page of the linked list. This means that there is an overhead of at least one additional I/O for large data types.

Some SQL commands can be used with TEXT and IMAGE data. INSERT, UPDATE, SELECT and DELETE can all be used, but operators such as "=", ">", "<", IS NULL and IS NOT NULL are not legal for long data type fields.

Remote Procedure Calls

Remote procedure calls (RPC) allow an application on one Sybase server (or client) to execute a stored procedure on another Sybase (or open) server. Stored procedures, a set of SQL commands created using Sybase's TRANSACT-SQL language, have been discussed above. Stored procedures enhance performance, since all of the commands can be executed with one call from the application program.

There is no support for a 2-phase commit with an RPC, since remote procedures are not within the scope of a Sybase transaction. This limits the usefulness of RPC's since if there's a failure in a trigger processing as part of an RPC, there is no notice returned to the originator of a failure.

Conclusion

Sybase is sitting in the best position of this industry. The company's growth rate is aggressive but manageable. The company has exhibited excellent technical and marketing management. The product has the top reputation. Sybase's principal business partners are Microsoft and Lotus. The June 1990 Software Digest, in the category of SQL Servers for OS/2, awards Sybase 3 stars, Oracle Server 2 stars, and both OS/2EE Data Manager and Gupta 1 star.

CHAPTER 6 - ORACLE

Introduction

Oracle Corp., is a leading provider of UNIX DBMS'. An Oracle database is portable to many different platforms as long as you maintain the same version of Oracle across all platforms. (At any given point in time you will not find the same version of Oracle on all its platforms.)

Oracle's primary advantage is multi-platform portability and networking. What other database software lets you run the same application on 50 different computer systems and share data between them to boot? This is Oracle's strongest selling point in large corporations.

Oracle's portability advantage is less than before because the other UNIX RDBMS' also have developed portability. However, Oracle does support Macintosh. Among its competitors, only Sybase has a similar capability. Finally, Oracle has put much effort into expanding its product and services and now it gets a major part of its revenue from selling add-on products and consulting services.

The time lag from when a new Oracle product is introduced for one environment and then is ported to another can be two years. In the past, products have appeared first for VAX/VMS, then migrated to UNIX and MS-DOS. It took the better part of a year for SQL*REPORT WRITER to make the transition to MS-DOS after being available on the VAX. This dilemma applies to any multi-platform software, but because Oracle runs on so many different computer systems, it can become a problem for users with heterogeneous environments.

The Oracle Engine

Oracle multi-threads in its handling of I/O, but uses a process-per-server architecture for other functions. For example, in the OS/2 world Oracle assigns each log-on an OS/2 process, at 300K each on the server, compared to 46K required by SQL Server's single-process threads. The up side of this architecture is that, within limits, it can use multiple processors. The downside is that it consumes memory resources and incurs extra CPU overhead as compared with a multi-threaded DBMS kernel.

Oracle 6.0, the newest version and the one that runs under OS/2, corrects many of the deficiencies of prior versions. It includes a new row-level locking feature that overcomes a major deficiency of previous Oracle versions that locked entire tables on updates. It is rumored, however, that it is this row level locking feature that prevents Oracle from running across VAX Clusters.

Version 6.0 also includes asynchronous I/O capabilities and improved data buffering.

There is an Oracle for OS/2 which runs on a number of different networks, including Novell's IPX/SPX, NetBIOS, and TCP/IP. SQL*Connect allows users to connect to remote Oracle databases. Oracle's distributed database technology enables users

to query local and remote Oracle databases within a single query. However, Oracle does not optimize distributed queries or support distributed updates. Oracle also has gateways to non-oracle DBMS' such as DB2, but in practice these gateways have had stability and performance problems.

Oracle supports a "row-level multiversioning" mode that is similar to SQLBase's read consistency or Interbase's multi-generational approach (similar in functional goal, different in implementation, since prior images are reconstructed from the journal). Read locks are not used; instead, read only queries see a consistent set of data based on the time stamp of the query's start. Updates never block read only queries. This, together with its row-level locking for tables and indexes, makes Oracle suitable for mixed reporting and transaction environments. This Oracle multiversion capability is not available, however, in distributed environments.

Oracle queries are sensitive to the SQL syntax used by the programmer. This means that an SQL command behaves differently depending on the order of the table names in the SQL WHERE clause. An unoptimized query can take several orders of magnitude longer to process; this increases as the query draws upon more relationships and more tables. This, of course, violates the whole premise of the relational model. Instead of using a cost-based algorithm to determine the best way to JOIN tables, Oracle puts the burden on the programmer. Skilled programmers may be able to find the right syntax, but if table sizes change, then the program should be modified. The situation can get worse when outside 4GL's or spreadsheets are used to access Oracle since then the generated syntax may be completely out of programmer control.

Oracle 6.0 has implemented a "poor man's stored procedures", OTEX, which is similar to SQLBase's chained-SQL. OTEX provides a performance boost in benchmarks like TP1, but is limited in practical value because it has no support for branch and control logic or any programming logic. Like chained SQL, OTEX can only return the error code of the last SQL command executed.

Tuning of the Oracle server must be done very carefully and only after one is familiar with how Oracle will respond to your settings. There are literally hundreds of different potential adjustments that can be made to Oracle that will affect its performance - the configuration, the initialization, program controls, rollback procedures, etc. Some of the tunable items are completely undocumented. A server that is fine-tuned for 24 stations will waste resources when 12 stations are running, and a server tuned to make the most of resources for 12 stations could easily malfunction if you attempt to run it with 13 or more. Worst of all, if you do happen to exceed the available memory of setting a parameter too high, no error message is given; Oracle simple behaves erratically.

In the hands of an experienced DBA, Oracle can be fast, principally because of an ability to take advantage of a large disk cache.

Oracle tools

SQL*Forms is Oracle's application development tool set. SQL*Forms doesn't support "IF-THEN-ELSE" or branching types of program logic. Oracle has promised enhancements in the form of Oracle's PL/SQL.

SQL*Forms is a good tool for its mid 80s heritage. By today's newest standards, however, it is not considered state of the art. There are no windows or part-screen forms; forms must be full-screen size; and only one form can be opened in data entry mode at a time. There is a lack of capability to cut and paste between forms.

SQL*Forms is more useful for forms based applications like data entry, than for general purpose transaction processing. You can't dynamically change properties of fields (i.e., make one field display only based on the value in another field), and you can't "parameterize" table names in SQL triggers. Finally, though it's possible to call SQL*Plus from SQL*Forms, it's hard to pass parameters, between different Oracle front-end tools.

These limitations point to the fact that there's no full-function fourth-generation language (4GL) in Oracle. This means that any transaction processing application that's beyond the scope of SQL*Plus (i.e. one with any conditional logic or error checking) and any forms application beyond SQL*Forms must be done in a programming language with embedded SQL. Although Oracle's embedded SQL precompilers are good, programming in a traditional language usually requires more time and effort than in a 4GL.

The limitations mentioned above mean that there is a need for outside developer tools. In spite of the vivid image of Oracle shooting down dBASE, Oracle isn't a dBASE competitor, but would rather benefit from cooperation with tools like dBASE. It's hard to understand, then, why Oracle spent money to publicly attack a product it couldn't replace.

Using any outside PC-specific language or front end unfortunately eliminates a principal Oracle advantage, portability. This portability is only possible if development is done exclusively with Oracle's tools. An all Oracle application will run virtually unchanged on more than 80 different computers and operating systems, including workstations, minis, and mainframes. Data on Oracle Server is also portable among XENIX, OS/2, VAX, and mainframe platforms.

For the PC world, Oracle Server has the reputation of being the hardest product to install and use. Learning to tune it is difficult, especially because of its lack of automated features, such as optimizers and dynamic allocation of RAM and by the need to do tuning off-line. By PC standards, Oracle is complex and demanding.

Conclusion

Oracle's strengths are its portability and large base of application support. At 1990 sales of about $1 billion it is the largest (by far) DBMS software company. Sometimes, when its products are compared negatively with competitors such as Sybase it's important to remember that Sybase has less than 10% of Oracle's sales!

Technologically, it lags behind its competitors, and it is a complex relational DBMS that is difficult to administer. The product consists of new features layered on an old architecture. It needs stored procedures and triggers. It should also implement either clustered indexes or hashed tables for performance. The

development tools need upgrades.

Speaking of documentation, none of the products reviewed in this report get very high marks compared to the type of documentation IBM provides with DB2. Based on a scale of 10 (best), the following was a consensus opinion of a few analysts:

Oracle	4
Sybase	5.5
Interbase	4
DB2	8.5

Oracle is the largest supplier of UNIX RDBMS'. Whatever technical advantage Oracle once enjoyed has vanished. The latest release of Oracle 6.0 is missing features such as stored procedures, triggers, BLOB data types, user defined data types and functions, disk mirroring, clustered indexes, a multi-threaded database architecture, cost-based optimization for distributed and nondistributed queries, and distributed updates.

In response to these deficiencies, Oracle has announced that Oracle 7.0 will include most of these capabilities when it ships. At this point in time and given Oracle's track record with previous promises a release 7.0 date for any given platform is unknown. Oracle version 6.0 was released two years late; Oracle's PL/SQL and SQL Forms 3.0 are both more than two years late. It wouldn't be unreasonable to expect the UNIX versions to ship in the late/91, early/92 time frame.

A potential user must also consider just how much effort and time it takes to optimize queries using Oracle's trial and error method. To tune Oracle, you must shut down and restart the server with every parameter change.

DEC's promotions for Rdb are now taking a serious chunk out of Oracle's sales on VAX. Rdb has become a good product. It has triggers, referential integrity and distributed updates. The run time version is free. The developer's version is cheaper than Oracle.

CHAPTER 7 - INTERBASE

Analysts who have evaluated the Interbase technology seem to be impressed - especially given the size of the organization. While not up to Ingres' standards, the product is considered a strong candidate for second place in the technology race (with Sybase).

Engine

Interbase's database engine technology is oriented toward distributed database and large object management. The engine is optimized for handling random, unpredictable queries, and is especially designed for fast and high quality performance in a decision support role. The engine has been designed for good performance on workstation platforms. Cognos resells the Interbase engine, but without benefit to Interbase (except financially) because Cognos doesn't advertise the source of their technology.

Interbase offers the user a choice of a full multithreaded, central server approach (like Sybase), or a multiprocess approach (like Oracle).

While Interbase supports a 2-phase commit for distributed update, it doesn't have a cost optimized distributed JOIN (neither does Sybase). Neither Interbase nor Sybase have a distributed data dictionary. Interbase's capabilities in the area of heterogeneous foreign DBMS gateways is significantly less than what is available in other products such as Ingres, Focus or Sybase.

Interbase doesn't support a clustered index "database speed-up" technology like Sybase, but has a comparable or superior alternative technology known as "bit-map" technology. This approach uses bit vectors to represent whether a data field has or hasn't the values searched. Boolean operations are performed on the bit vector - a very fast process. I have run across this technology before and have found it to generally well regarded.

Interbase doesn't support the distributed DBMS requirements for fragmentation or data replicates. Neither Oracle or Sybase supports this functionality.

Interbase has direct support for SMP (Symmetric MultiProcessing) and can take advantage of several parallel processors under the same skin (with an appropriate operating system). These processors can be tightly or loosely coupled. Interbase can, then take advantage of VAX Clusters, which neither Sybase or Oracle can use to full advantage.

Disk mirroring is supported through the process called "shadowing". This mirroring capability is also supported for CPU's and both of these technologies are useful in situations requiring non stop operation.

A unique capability is Interbase's support for application specific functions. This capability allows a user to easily extend the range of database commands by adding new functions coded in C to the DBMS kernel. This facility is helpful in the

manipulation of BLOB data.

Event Alerters

Interbase has added event alerters with Version 3.0. An event alerter is a signal sent by the database to waiting programs to indicate that a database change has been committed. Event alerters work remotely, even across multi-vendor networks. No other company has event alerters at this time. I mentioned to Jim Starkey that it would seem to be a simple functional addition to add event alerters to a system that supported the concept of triggers. He pointed out that implementation of the technology is made difficult by the need to support an asynchronous, heterogeneous environment.

Event alerters are a type of technology comparable to stored procedures and triggers, both of which Interbase supports. Interbase has no limit on the cascades that can descend from a trigger. In this whole functional area the Interbase technology is equal or superior to that offered by Sybase.

Event alerters offer the following benefits:

* No network traffic or CPU cycles are consumed by the waiting program.

* Notification is effectively instantaneous, not dependent on some polling interval.

* Event notification works remotely, even across differing platforms. The notification mechanism is managed by Interbase.

* Unlike a trigger, an event alerter can affect programs outside the database.

BLOB data types

Interbase includes a BLOB data type (binary large object bin). A BLOB has no size limit and can include unstructured non-relational types of data such as text, images, graphics, and digitized voice. Interbase handles a BLOB as a single field in a record, like a name, date, or floating point number. It can then be governed by concurrency and transaction control.

The ability to create "database macros" which can be executed by the database engine is supported within Interbase (BLOB filters). A BLOB filter is a centrally stored, user written procedure that tells the database system how to translate BLOB data to another format. Because they are stored in one place and managed by the database, BLOB filters are simpler to create and maintain than similar code in an application. BLOB filters are an area in which Interbase is ahead of its competition.

Interbase has array support for arrays of up to 16 dimensions in the database. Arrays are stored as a single field in a record, so retrieval is expedited. Array elements may be any Interbase data type except BLOBs and other arrays. Arrays are widely used in scientific processing and are very expensive to normalize for a relational DBMS. Normalization of an array normally means creating much added

redundant data to generate separate records for information that is really only different at the field level.

Multi-generational system

Jim Starkey was first exposed to the idea of maintaining multiple generations of database records by reviewing work done at Prime Computer. Subsequently, he pursued these ideas further at DEC and finally founded Interbase as a DBMS company which to create and market software embodying this idea. As implemented at Interbase, this technology offers good functionality for concurrency control and the ability to maintain consistent database views for multiple readers.

When a database management systems uses locks to maintain consistency, its concurrency control can be a two-edged sword. As the system protects transactions from conflicts, it causes them to wait for each other. Such an approach leads to deadlocks that force transaction rollback.

Interbase maintains data consistency through the use of "multi-generational" records. When a transaction modifies or erases a record, Interbase creates a new record version instead of overwriting the old record. In most cases, the old record version contains a compact record of the changes.

The versions are chained together to form a multi-generational record. When a transaction starts, it reads the most current version. Thus, a read transaction is never blocked. For example, when a report program reports on the state of the database, it ignores changes that were made after it started, so other updates proceed unhindered. The reader, then, always gets a consistent view of the data base correlating to start time of the transaction. Most other DBMS products provide a view of the current state of the database.

This multi-generation approach obviates the need for Interbase to implement "snapshots" since the base engine's technology offers a functional superior alternative to what benefit shapshots provide. Neither Sybase or Oracle have working implementations of snapshots.

If there has been an update in a portion of the database that happened after another writer's start time (a collision), the DBMS must roll back anything done by the second writer and give it a new time stamp that allows it to have a consistent database view. Interbase claims that this process is actually superior (performance) to other types of locking schemes. It was outside of the scope of this study to verify this claim, but I think it represents a key issue.

Interbase management confirmed that there is an overhead attached to processing required by this multi-generational approach that must continually be paid. Interbase doesn't run TP1 benchmarks and it was the estimate of both Richard Finkelstein and Herb Edelstein (both of whom were consulted as part of this study) that Interbase would not do well on these types of benchmarks. This is a problem that Interbase management must address if there are plans to port to the OS/2 environment. It's a problem because the marketing image of a fast TP1 number is essential for competing in this market segment (or for that matter in the commercial

end of the UNIX market).

Interbase's multi-generational approach also deals with other database management issues:

* All DBMS systems must maintain a copy of the previous state of a record in case the transaction aborts or rolls back. The traditional approach involves a "before image" journal, a separate file into which the system copies the old version before overwriting the record in the database.

* However, Interbase uses the database itself as a before image journal through its multi-generational records. Interbase management claims an advantage for this approach because 1) multi-generational records require less I/O than separate before image journals, 2) no separate recovery program is needed, and 3) recovery is instantaneous as soon as the machine recovers from a crash, with the database available for use.

Interbase tools

A number of VAX tools that are DSRI compatible operate with Interbase (eg Smartstar, Powerhouse). I didn't pursue this subject in great detail as part of this study, as I relied on Interbase management's discussion of their capabilities. It was their assertion that this is an area of weakness as compared with either Oracle or Sybase. The company has relied (not unreasonably) on compatibility with DEC's DSRI specification to take advantage of the fact that any tool running on Rdb will run on Interbase. This type of tool support, however, doesn't help in heterogeneous environments, an area of Interbase engine excellence. According to Starkey, it is the paucity of multiplatform tools that has been the principal reason for the company's decision to shy away from commercial markets so far. The next Interbase release version will concentrate on improving Interbase's own toolset.

The combination of a dBASE front end environment (with its millions of users) and a powerful, distributed server back end from Ashton Tate would certainly spike market interest in client server style computing. I am convinced that client server database oriented approaches are the most promising way of attacking cooperative processing.

Conclusion

In a review of conversations that were held with aerospace/engineering Interbase customers I determined that the product is well liked and is considered to perform well in distributed environments. The development tools were liked and support from the company was fair to good.

The company's management stated that their marketing focus was toward 5 vertical markets:

Manufacturing

Finance
Engineering/scientific
Network management
Aerospace

As management analyzes the decision of moving Interbase toward becoming a player in the general commercial client/server business, it should not underestimate the amount of effort to successfully accomplish this move. I think that this effort is likely to be more a management than a technical challenge. Interbase has no visibility in commercial client/server markets. Interbase's management, especially marketing, is likely to have to change significantly for such a metamorphasis to work.

CHAPTER 8 - QUICK REVIEW OF OTHER COMPANIES

Gupta Technologies
SQLBase
1020 Marsh Road
Menlo Park, CA 94025
415-321-9500

SQLBase is missing some advanced relational capabilities like referential integrity, stored procedures and triggers. It is capable of storing and executing a precompiled set of SQL commands without branching, error checking, or program control. This is called chained SQL - a kind of poor man's stored procedures.

SQL Windows is an excellent programmers tool for developing sophisticated windows based applications. It is not too easy to learn but it very capable. Gupta has developed or is developing links between SQL Windows and other DBMS' including OS/2EE, Oracle and Sybase.

There is a DOS version of SQLBase. All of Gupta's products will appeal to the developers who are from the PC world, since they carry a PC, rather than a minicomputer flavor.

A big problem with SQLBase is poor and missing documentation. Gupta's software has had more quality problems than is normally considered acceptable in mainframe environments.

Gupta is rapidly growing and appears financially successful. Novell just purchased 20% of the company. Gupta has about the same number of employees as Interbase.

Ingres Corporation
INGRES
1080 Marina Village Pkwy
Alameda, CA 94501
415-769-1400

Ingres comes with a multi-threaded, multi-server architecture. Ingres has the best cost-based software optimizer technology available today. Its optimizer stores database statistics and usage histograms.

Ingres has a query flattening algorithm that levels out different SQL syntaxes with the same semantics to make sure that they are interpreted identically and run with an optimal path. In this way, Ingres is opposite from Oracle.

Ingres comes with a complete and high quality application development tool set called Application by Forms.

Ingres at this time supports multi-site updating with a programmer controlled 2-phase commit protocol.

Ingres' "Knowledge Management Extension" (KME) allows users to store triggers in the DBMS catalog. This can be used for protecting domain integrity or for centrally implementing referential integrity and business rules.

Most independent analysts agree that Ingres technology is superior to all of its competitors. In addition, most analysts like the company since it has a generally good reputation of not exaggerating or lying about capabilities.

Unfortunately, Ingres has been bedeviled by less than professional top management. The marketing and general management capabilities of the company have been suspect. Ingres' recent acquisition by ASK Computer Systems is not a good sign; I know of no cases where an application vendor has successfully acquired a DBMS and tools company.

Progress Software Corporation
PROGRESS
5 Oak Park
Bedford, MA 01730
617-275-4500

Progress has been well accepted in the VAR and small organization developer community. It has a complete DBMS capability and 4GL. All applications in Progress must be written in its own 4GL language since it does not support an API for languages like C or COBOL. A DOS version is available, in addition to versions for dozens of UNIX platforms and the VAX.

Progress Release 5 has a multi-threaded, multi-server architecture similar to Ingres.

The heart of Progress is its 4GL, if you like the 4GL product, you'll like Progress. Since the package is sold with 4GL and database bundled it is very competitively priced compared to its competitors. Based on employee count, the company appears to be twice the size of Interbase.

XDB Software
XDB
7309 Baltimore Avenue
College Park, MD 20740
301-779-5486

XDB's principal importance in the market place is as a PC development platform for IBM's mainframe DB2, DBMS with which it is highly compatible. XDB not only duplicates DB2 SQL syntax, it duplicates the messages and return/error codes. In

addition, XDB displays data in the same way than DB2 does and it maintains DB2 SQL restrictions. The combination of Micro Focus' COBOL with XDB makes an ideal PC development platform for mainframe applications.

Informix Software, Inc.
Informix
4100 Bohannon Drive
Menlo Park, CA 94025
415-322-4100

Informix is one of the most popular DBMS' for UNIX environment. Its price and low hardware requirements have made it very popular among VARs. Informix-4GL is a good development tool with good performance. Informix has been maintaining its database technologies successfully with new releases that include disk mirroring, on-line database backups and parallel I/O operations. In addition, Informix supports BLOB data types. Informix's distributed system has a cost-based optimizer which takes into account data location, and both communication and hardware cost.

END