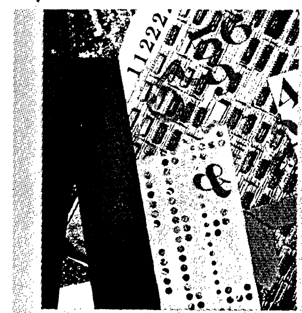


DBMS

FOR INFORMATION CALL
NORM ZARESKI
BRITTON LEE, INC.
111 N. SEPULVEDA BLVD.
SUITE 250
MANHATTAN BEACH, CA 90266
(213) 318-7009

# INTELLIGENT DATABASE MACHINES

By Philip A. Naecker

## Britton Lee's Hardware Approach to Database Management Can Lead to Efficient and Host-independent Database Access. Part 1.

*Relational databases have received more press than just about any other software topic lately, except possibly networking standards. Britton Lee, Inc. (BLI), Los Gatos, California, markets intelligent database machines and relational servers that combine both of those hot topics and add the lure of data sharing, high-performance computing on PCs, and possibilities for excellent price-performance ratios for database operations. Britton Lee has taken the flexible approach to databases, allowing the user to mix-and-match the language, host machine, and performance to match the application. And, for some users, it looks like the Britton Lee approach might just be the ticket to efficient and host-independent database access. In Part 1, we'll examine some of the issues involved in special-purpose vs. general-purpose processing, look inside the BLI architecture, and explore the options that database hardware offers.*

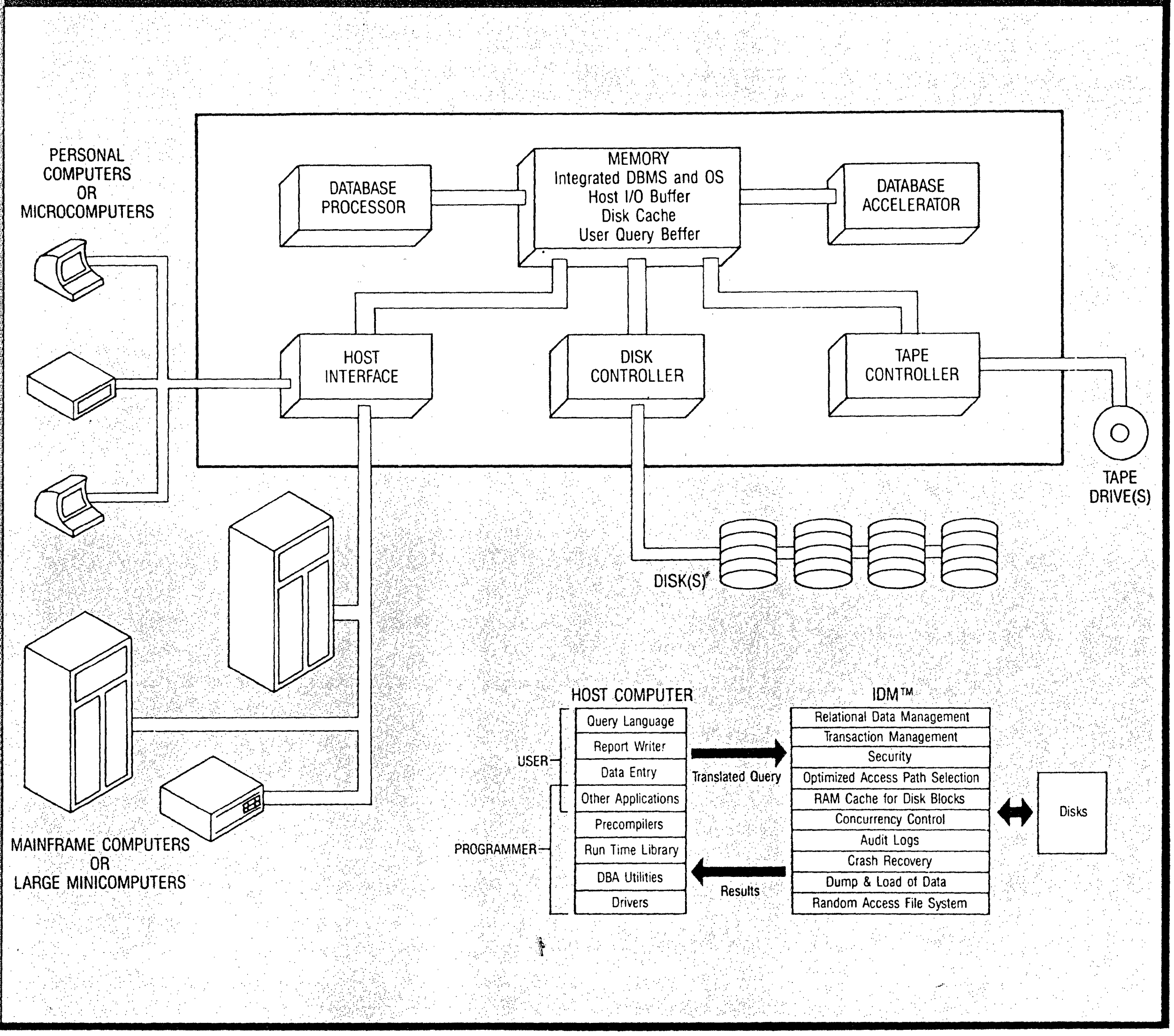
Much of the press addressing relational databases has revolved around "The Great Performance Debate." On one side are system managers and some MIS managers, who argue that the unreasonably large amounts of resources are consumed by applications built using relational database software running on a general-purpose computer. On the other side are application developers and database gurus. In the first place, they argue, it's better that the machine do the work than the programmer, and what you need to look at is total productivity—value for your dollar. Because of the inherent flexibility of relational databases, they say, there are tremendous opportunities for reducing software development and maintenance costs by using the relational database approach.

To deal with the "performance problem," real or imagined, and to provide rapid and uniform response time for on-line processing, there are two strong players in the database machine market. Teradata Corporation, Los Angeles, is considered the Big Gun. They build very big, very costly, highly parallel machines for time-critical applications.

DEC PROFESSIONAL

Reprinted from *DEC PROFESSIONAL*, Vol. 5, No. 10, October 1986, and Vol. 5, No. 11, November 1986, Professional Press Inc., 921 Bethlehem Pike, Spring House, PA 19477, (215) 542-7008. November 1986.

FIGURE 1.



***The Britton Lee database architecture.***

Teradata machines run as a back end for IBM mainframes, and are typically found in such time-critical database applications as on-line credit card validation for major banks. They can handle many gigabytes of on-line storage and hundreds of complete database transactions per second, and cost more than most VAX installations.

Britton Lee is taking a lower-end approach, building small- to medium-scale machines with capacity for a few hundred megabytes to a few gigabytes of storage and processing capacity in the range of five or 10 transactions per second, up to perhaps 20 per second if the transactions are very simple and use indexes efficiently. The Britton Lee machines are much cheaper, though, starting at around $50,000 for a complete machine from the Relational Server 300 line. The Britton Lee boxes connect easily with VAXs; indeed, most of BLI's installed base of about 600 machines has at least one VAX connected to the database machine.

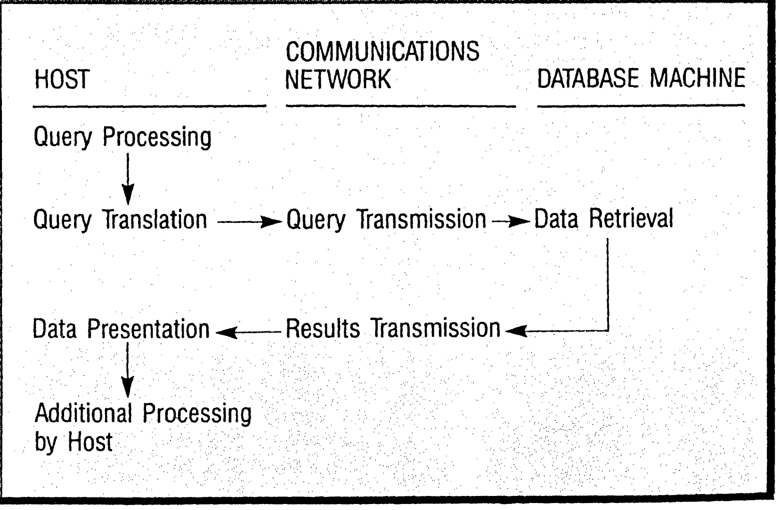
## The BLI Architecture

The Britton Lee approach to relational databases is simple: Database operations are relatively simple I/O and computational tasks. They do not need the support and consequent overhead of a generalized computing architecture. In the case of the VAX, there are some three hundred instructions, a dozen or so addressing mechanisms, and an architecture to support a virtually unlimited number of I/O device types (Figure 1).

While this makes the VAX a wonderful place to develop software, Britton Lee and others who promote the database machine concept feel they can avoid the associated overhead and do the same amount of work as a VAX-11/780 with a Z-8000 processor and a few pieces of specialized hardware. In fact, the BLI benchmarks put some models of its database machines at several times the performance of a dedicated 11/780.

In the BLI database architecture, there are distinct phases in database access (Figure 2).

FIGURE 2.



*Phases of database access.*

TABLE 1.

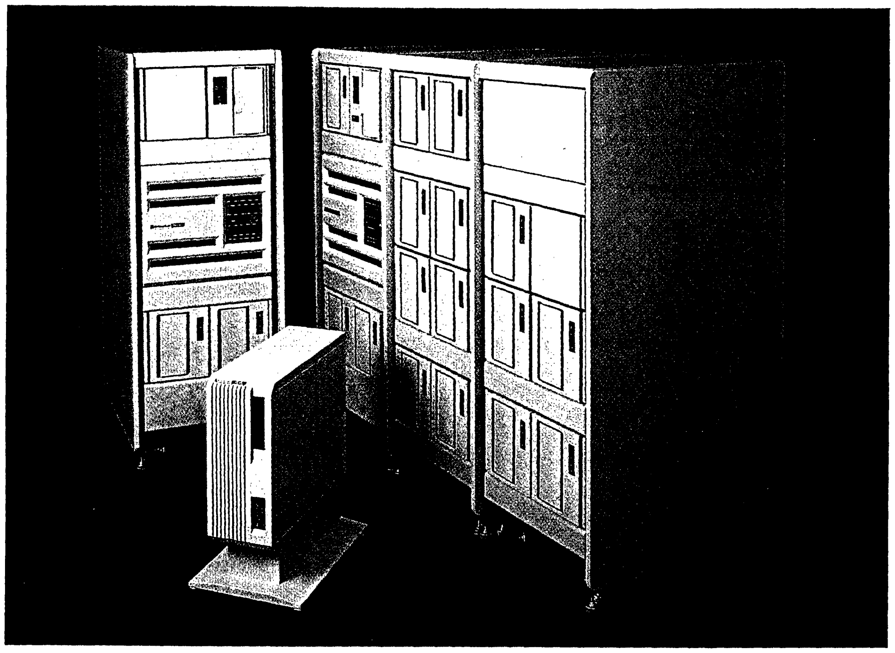
| |
|---|
| VAXs running VMS, ULTRIX, System V UNIX, BSD.4.2 or 4.3 |
| AT&T 3Bs |
| PCs running PC-DOS and MS-DOS |
| IBM 43XXs and 30XXs running VM/CMS |
| Apollo |
| Pyramid |
| Sperry 1100s |

*Supported hosts for BLI database machines.*

In the first phase, a query is constructed, either by accepting interactive input from a user in a database query language such as SQL or QUEL, or by operating on SQL statements imbedded in a third-generation language such as FORTRAN or COBOL. This phase takes place on the host machine, which may be any of a long list of processors and operating systems, from VAXs to IBM mainframes to PCs (Table 1).

The host also translates the query into a standard message format, and then transmits it to the database machine via a standard communications protocol. In keeping with its stated goal of providing database machine access to all users in an organization, BLI supports four different communications protocols (RS232, IEEE-488, IBM Channel interfaces, and XNS Ethernet). So, it's a good bet that if you're a VAX user you'll have no trouble communicating with the database machine. The host software also translates any data in the query (say, numbers or text strings) into a standard format so that the database machine doesn't have to know what kind of host it's communicating with.

At this point, the database machine takes over. Since the query already has been parsed and translated into a standard format, the work that's left to be done is quite well defined, if not simple. The BLI machines run an improved implementation of the standard relational database software (first developed in the public domain at U.C. Berkeley in the early 1970s) similar to many software packages available on VAXs. BLI also gives the software a hardware boost by using specialized disk controllers, a special on-disk format for the data (after all, they only need to store one kind of file on the disk!), and a hardware accelerator they call a "search engine." The search engine helps search indexes or search when the item of interest is not rapidly available via the index structure that the relational database software maintains. The database machine software also does many frequently used numerical operations, such as calculate averages, count items, and



*Britton Lee's "family" of relational database management systems.*

the like, offloading these tasks from the host as well.

While the query is being processed on the database machine, the host is free to do whatever it wants. In a general timesharing environment like a VAX running VMS, this means that the host CPU can be busy doing the work of other users on the system. This contrasts with the software-only approach to databases, where the software system does the database retrieval, disk accesses, record locking, and other tasks associated with database operations.

When the database machine is finished processing the query, it returns the results, en masse, to the host. Actually, it returns the results in a standard relational form, called a table, which is how the relational software running on the host probably wants to see it anyway. In the last processing phase, it's up to the host software to convert the table into whatever format the user might want, which could be displaying it on the screen, putting it into a file, feeding it to a program one record at a time, or even putting it into a *Lotus* spreadsheet. The host software also translates from the standard BLI data format into whatever data format the host requires, thereby completing the process.

## Special Purpose or General Purpose?

The initial ballyhoo about database machines was the potential for vastly improved price-performance when compared to a general purpose computer like a VAX. But, price-performance calculations are full of unspoken assumptions and real-world complexities.

There is the potential for significant price-performance gains because of the application of special-purpose software to the database problem. It stands to reason that a specially designed, single-purpose computer can do a better job of a single task than can a more complex, multipurpose machine. So, if you have to expand your hardware anyway, why not add a few special-purpose processors instead of another general-purpose timesharing machine?

In some cases, this argument may be enough to carry the decision. If you have a database load that fits nicely onto one of the many BLI boxes available, you may be able to just pick up your application and put it on the database machine. More likely, a new application may come along, and the expansion decision is between upgrading your VAX or expanding your VAXcluster and adding a database machine for the new load. Indeed, BLI finds that most of its customers are buying the machines for new applications, rather than to port existing applications to the higher-performance environment.

The host software (to do the query processing and translation phases) takes very little host CPU time, so it's not likely to be a big load on the host unless you have very, very many simultaneous accessors to the database machine. Even then, users on many different machines can all communicate with the same database machine to access the same database, so you can distribute your host load over several VAXs or a combination of VAXs and other machines, including PCs.

If your software development staff is familiar with and happy on a VAX, they don't really have to learn much about the database machine. Since you do all the software development on the VAX (or other host), you still can use the VAX debugger, software development tools, and editors to build the software. But, when the system actually starts accessing data in a production mode, most of the work is done on the database machine, and the VAX has a substantially reduced workload when compared to a software-only approach in which database software on the VAX does the job of managing the database.

On top of all this, some applications may be very response-critical. A telephone directory application, for example, may demand response times of less than two seconds. To get this kind of response time with a software rela-

The real strength of the BLI architecture is its tremendous egalitarian attitude toward data.

tional database product running on a general-purpose computer may require the purchase of a very large processor. It would be largely idle, however, because the transaction volume doesn't keep it busy all the time. A database machine might be a considerable price/performance win in this sort of circumstance, because the special-purpose hardware can provide more rapid response to queries.

Of course, there is a down side to the purchase of a special-purpose machine. The most serious drawback is the same as its major advantage, oddly enough: it's a special-purpose machine and it only can do one thing — process relational database requests. If you run out of cycles on your host VAX, you still are going to have to upgrade it—you won't be able to offload any non-database work to the database machine. This contrasts with the solution of, say, adding another 8600 to your cluster, where the added machine can be used to satisfy any and all processing requirements, not just a certain class of problem.

Another problem is that the database machine, although it is a special-purpose computer, still is a computer with an operating system, utilities, several different kinds of hardware and all the headaches attendant to those components. For example, don't expect to find VMS BACKUP on a BLI database machine — there is a special utility for backing up databases, and someone is going to have to learn how to use it. The hardware is mostly standard components (Eagle drives and the like) and DEC Field Service is available.

And, speaking of fixing things, there is the problem of redundancy. One of the beauties of VAXclusters is that you can keep critical applications running when a processor fails, because a VAX is a VAX is a VAX — you can run most applications on any of the processors in the cluster. But, if you have only one database machine, and that fails . . . You might want to take part of the dollars you saved with the price/performance win you got when you decided on the database machine, and plow it back into a little redundancy in your database processors. Even so, the disk drives on a database machine aren't connected to a common interface like an HSC, so you'll have to do some cable swapping or at least some fancy software footwork on the host to get the right database up on the right machine after a hardware failure. (One good thing here — some of the database machines provide built-in disk shadowing, so at least you can pay extra bucks for additional drives and have protection from disk crashes.)

To sum up the price/performance issue, if you can't keep the special-purpose machine busy a substantial amount of time doing things that would otherwise be done on the host, and by so doing avoid upgrading your host, then the argument that it's cheaper to get a special-purpose machine becomes a little thin. The down side (reliability/redundancy, additional operations overhead to maintain another machine, etc.) starts to add up.

But, don't give up hope of getting one of these little puppies just yet.

## Data Switch

The real strength of the BLI architecture is its tremendous egalitarian attitude toward data. A BLI database machine doesn't care who it's talking to (or, more specifically, what kind of host machine you're using). Besides your friendly VAX or MICROVAX, the BLI machine will talk to PCs running PC-DOS or MS-DOS, UNIX boxes, Big Blue boxes, and a whole host of minis. All of the hosts connected to the database machine can access the same database, at the same time, in a shared read-write mode. When you consider that it wasn't until V4 of VMS that you could have multi-CPU shared read-write access to a file on an HSC in a cluster, you begin to see how impressive an achievement this really is. The VMS folks gave us multi-CPU access in a pretty homogeneous environment (you can't even have a PDP-11 in there playing ball). But, the BLI folks have linked machines with completely different architectures and data formats, not to mention manufacturers.

Now, you may be one of those lucky folks who doesn't have anything in the computer room that doesn't say "VAX" on the front. But, before you let that smile on your face grow too broad, you might take a look beyond the computer room. Have you ever heard one of your PC users say to you, "Can you help me download the data from the VAX to my *Lotus* spreadsheet?" Or, has the VP Finance come to you and said, "I want this new Corporate Budgeting application to be available from both the IBM and the VAX. And, oh yeah — I want to update the information using my IBM PC and Joe's Rainbow. Can you do it?"

With a BLI machine, you can use your existing Ethernet or RS-232 connections to access data on the database machine from almost any of the other machines in the organization. Using existing software, you can upload parts of databases into *Lotus* or *Multiplan* spreadsheets, RMS files, or provide it to a host language program running on any of the processors connected to the box. And, you can update the database from any of the connected hosts too, provided the user has the appropriate access clearance on the BLI machine. Pretty slick, huh?

Well, BLI thinks this is a pretty slick thing, too, and they intend to play it for all it's worth. New software products are on their way, emphasizing the connectivity of the BLI boxes and providing new, easier-to-use interfaces for users to get to their own data, probably from the PC they have sitting on their desks. There also are new software products to aid in the development of full-blown applications, running on the VAX with the database providing back-end database management.

Not to take anything away from BLI, I should add that there are beginning to be some software-only products in the VAX market that hold the promise of data sharing between dissimilar machines. Two that come to mind immediately are *INGRES* from Relational Technology, Berkeley, California (See *DEC PROFESSIONAL*, Volume 5 Number 8, August, 1986 p. 68.), and *gds/GALAXY* from Groton Database Systems, Groton, New Hampshire (watch for our review in a future issue of *DEC PROFESSIONAL*). *INGRES* just announced *INGRES/STAR*, a product that they say will allow a single database to exist homogeneously on dissimilar computers. And, the just-released, *gds/GALAXY*, certainly appears to have all the bases covered in distribution across VAXs, UNIX boxes, and Sun and Apollo workstations. Although I haven't done a detailed review, *gds/GALAXY* has the apparent advantage of being designed from the ground up with distribution across dissimilar machines built in, whereas *INGRES* has been around for awhile without that capability. But, both of these are software solutions, whereas the BLI product is a hardware solution.

## TABLE 2.

| | IDM500X | IDM500XL | IDM500XLE |
|---|---|---|---|
| Typical Minimum Data Storage (MB) | 100-1200 | 150-1200 | 300-1200 |
| Minimum Concurrent Queries* | 18 | 33 | 74 |
| Simultaneous Users | 250 | 400 | 500 |
| RAM(MB) | 2 | 3 | 4 |

* For queries of maximum complexity.

***Intelligent Database Machine 500 Series models.***

## TABLE 3.

| | |
|---|---|
| RAM (MB) | 6 |
| Disks | 16 Drives |
| Controllers | 4 SMDE |
| Communication Interfaces | RS-232<br>XNS Ethernet<br>IEEE-488<br>IBM block multiplexor |
| Special Hardware | Database Search Engine (8 MIPs) |
| Cost | $100-$300k |

***High-end BLI database machine — Model IDM500XLE.***

## The Options

Let's take a look now at the options available from BLI. Just like the menu in a Chinese restaurant, there are more combinations of entrees and side dishes than you may care to calculate. We'll look first at the main course, the database machine hardware. Then we'll look at the side dishes that make the meal really tasty.

For the database machine itself, there are two families — the low-end Relational Server 300s and the high-end Intelligent Database Machine 500s (IDMs). The IDMs were out first, hitting the street in 1981. There are now three basic versions of the IDMs: the IDM500X, 500XL and 500XLE. (Sound a little like Mercedes models, don't they? Do you think that was on purpose?) All three offer large disks and potential for connection to a large number of hosts. They also have the horsepower to support numerous simultaneous connec-

One nifty thing that all the IDM models offer is disk mirroring.

tions and a large number of simultaneous users. See Table 2 for the details of the three base configurations. Basically, each of the models can be upgraded to the top of the line, which is described in Table 3.

The search engine available on the high-end models is specifically geared toward doing string searches and similar activities on a large number of records. This can be used when the retrieval of information is not based on an index. Indexes can be expensive to maintain if they are on large data items (like names of chemical compounds), and of less help if you are doing a search that might return a large number of the records in the database. Further, indexes on strings are more difficult to process when used in retrieval for wildcarded values such as "*methane*". The database search engine is just the ticket for this sort of pathological case. The search engine also gives a hardware assist to sorting.

There also are a number of fancy options you can get for your IDM: Reel-to-reel nine-track tape and *Megatape* high density tape (see related article on page 158), optional SMDE disk controllers, and of course, a wide variety of communications interfaces. One nifty thing that all the IDM models offer is disk mirroring. This is widely considered the best, if not only way, to protect valuable data from disk crashes. In a mirrored disk configuration, the hardware maintains two exact copies of the data. If at any time the two don't agree, the software figures out which one is right and takes the failing disk off-line. Users familiar with the VAX/HSC cooperative disk shadowing facility will find this concept familiar. However, the BLI mirroring facility does not support an on-line catchup process at this time.

The Relational Server 300 family is the newer, but lower-end, addition to the BLI line. Unlike the IDM family, the RS boxes operate in an office environment, not just a controlled computer room environment. The RSs also use the Small Computer Systems Interface (SCSI) internally, allowing BLI more flexibility in the packaging and design of hardware. Using a six-slot MultiBus card cage, the basic RS310 has a meg of memory, an 86-MB Winchester and a 60-MB cartridge tape. You also can purchase an eight-channel RS232 interface to supplement the standard XNS Ethernet LAN interface.

But, the really great part of the RS300 series is that it is 100 percent software compatible with the older IDMs. Actually, when you think about it, that fact is just a natural outgrowth of the whole BLI architecture, which is one of the things that makes the approach attractive. A user on a PC, or a VAX on the Ethernet, can connect with either the RS or the IDM and perform database operations with identical software on the host processor, using identical commands.

In Part 2, we'll examine host software, distributed processing, and redundancy in a BLI database machine environment.

*Philip A. Naecker is a Southern California-based consulting software engineer.*

*Britton Lee Intelligent Database Machines*
*Britton Lee, Inc.*
*14600 Winchester Blvd.*
*Los Gatos, CA 95030*
*(408) 378-7000*
*Price: $50,000 (Relational Series 310) to $300,000 (IDM 500XLE).*

# INTELLIGENT DATABASE MACHINES

By Philip A. Naecker

**Britton Lee's Hardware Approach To Database Management Can Lead To Efficient And Host-Independent Database Access. Part 2.**

*In Part 1, we discussed the issue of special-purpose versus general-purpose database solutions, the architecture of the Britton Lee, Inc. (BLI) database machine, and the options available using this hardware approach. Now, we will look at host software, distributed processing, and redundancy.*

The BLI database machines don't really have a user interface. The functions of query processing and display of results are left to the host computer, which communicates with the database machine in a proprietary protocol packaged within a public communications protocol, such as IEE-488 or XNS. The term "host" may be misleading, though, since it can refer merely to a low-end PC that allows users to compile requests in a language such as SQL, or using one of the several end-user query generators.

At its simplest level (if not the simplest to use), you can query a BLI database machine by constructing a query in SQL or IDL (a BLI variant of QUEL, and yet another SQL-like query language for relational databases). For an application developer or experienced database user, this is a perfectly acceptable, and perhaps even efficient, means of accessing data. IDL and SQL are supported on all hosts that interface with the database machines. BLI states that its SQL is compliant with the recently proposed ANSI Standard for SQL.

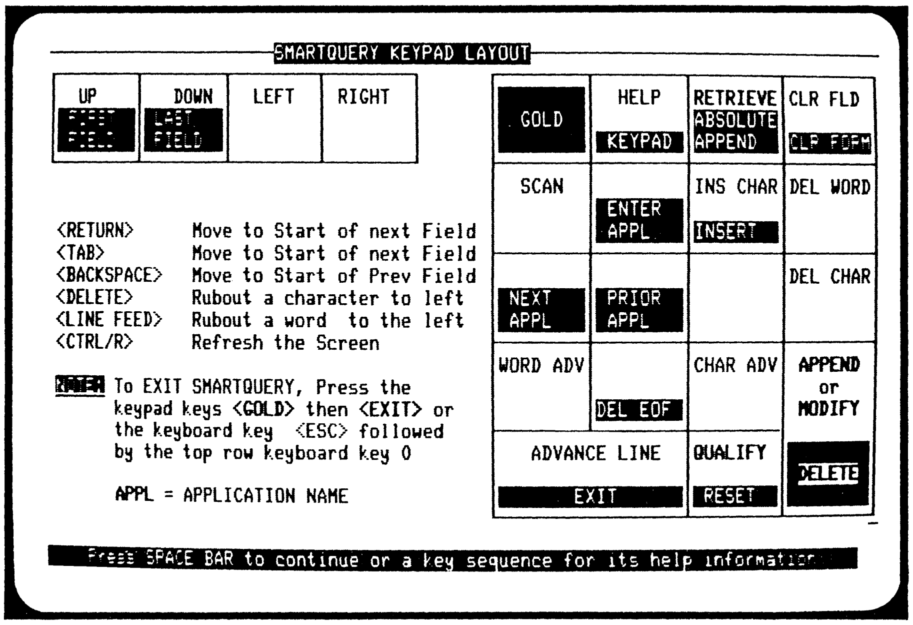
Third-generation language programmers also can write programs with imbedded SQL or IDL code. These programs then are precompiled to use subroutine calls to a standard BLI run-time library, and passed to the host language compiler. Alternately, the programmer can make calls to the run-time library directly. Not every host and operating system supports all the languages, but on the VAX, there is a C precompiler, and a FORTRAN precompiler is said to be on the way.

There also is a version of *OMNIBASE*, available from Signal Technology, Goleta, California that uses the BLI machines as a database back end. *OMNIBASE* is an application development package designed to aid users in managing data, and as a run-time library for programmers to call for data manipulation.

*OMNIBASE* provides a number of ways to develop applications that access the database machines, including a pair of components called *SmartDesign* and *SmartQuery*. *SmartDesign* lets you build a form using an FMS-like forms editor, then create a database directly from that form. It's about as easy as falling off a log. Then, using *SmartQuery*, you or someone else can operate on that database — adding, deleting, modifying, and retrieving data. Both systems are fully supported by online help that's more than adequate for most casual users, as well as for those with significant experience.

With *SmartQuery*, you can do a fill-in-the-blank type of query, including Boolean

*The* SmartQuery *Keypad functions for VT100- and VT200-series terminals.*



```
SMARTQUERY KEYPAD LAYOUT

UP           DOWN         LEFT         RIGHT        GOLD         HELP         RETRIEVE     CLR FLD
FIRST FIELD  LAST FIELD                                          KEYPAD       ABSOLUTE     CLR FORM
                                                                              APPEND

                                                    SCAN         ENTER APPL   INS CHAR     DEL WORD
                                                                              INSERT
<RETURN>      Move to Start of next Field
<TAB>         Move to Start of next Field           NEXT APPL    PRIOR APPL                DEL CHAR
<BACKSPACE>   Move to Start of Prev Field
<DELETE>      Rubout a character to left            WORD ADV     DEL EOF      CHAR ADV     APPEND
<LINE FEED>   Rubout a word  to the left                                                   or
<CTRL/R>      Refresh the Screen                                                           MODIFY

NOTE To EXIT SMARTQUERY, Press the                  ADVANCE LINE              QUALIFY      DELETE
     keypad keys <GOLD> then <EXIT> or              EXIT                      RESET
     the keyboard key  <ESC> followed
     by the top row keyboard key 0

     APPL = APPLICATION NAME

Press SPACE BAR to continue or a key sequence for its help information
```

logic statements like "greater than," "less than," and "not equal to." You can create such queries using the keypad on your VT100 or VT200 terminal, touching the main keyboard only to input search elements, such as names or numbers (see Figure 1).

You're not going to build full-blown, disk-crunching relational database applications using these tools, but they certainly give non-sophisticates access to relational techniques. Of course, databases created using *OMNIBASE* also can be accessed via any of the other host software tools available for the database machines, and vice versa. I used *SmartDesign* and *SmartQuery*, and found them to be consistent. They are a reasonable compromise between power and ease of use.

Also available on the VAX, is a tool called *Freeform*, from Dimension Software Systems, Inc., Grand Prairie, Texas. *Freeform* is more of an application generator, and not quite as geared to the end-user. But it has powerful retrieval methods, and provides good assistance for the user trying to do a sophisticated query. Besides running on VAXs under VMS, *Freeform* also is available under UNIX and PC-DOS.

BLI has the PC software market well covered. Another product called *PC/SQL*, from Boulder, Colorado-based Micro Decisionware, has been out for over a year, providing users with query access to relational databases on the BLI database machine. BLI co-markets *PC/SQL*. Since virtually every one of its sites has at least one PC, the market seems significantly large.

*PC/SQL* walks the user through the creation of SQL statements to retrieve information from the database machine. Using a windowing scheme close to the familiar "index card" paradigm, the user walks through a tree of questions to build a valid series of SQL statements. Once the query is complete, a keystroke sends the results off to the database machine. The machine returns the results as a single relational table that the user can view, or convert to another, more useful format. Included with the software, is the capability to convert the results to several standard spreadsheet formats (*SYLK*, *Lotus*, etc.), as well as the capability to form a coherent display of the results on the PC screen.

As of today, there is no update capability included with *PC/SQL*. This is not unlike a number of other end-user tools for relational databases. The problem with updating, is that there is a lot less room for ambiguity in the specification of the records of interest. After all, if you perform a query and get a few extra records or fields, you probably won't care. But, if you access a few extra records for an update of your shipping database, you might get some nasty calls from customers asking why their bills are so large this month! Of course, the BLI databases support all the traditional access protections to prevent unauthorized users from modifying the database, but that's not the issue here. In a friendly end-user database access tool, it's tough to make sure that a user who has the right to modify the database, performs only the modifications intended.

*PC/SQL* has a number of other convenient features. First, you can select the database machine from any on the XNS network, or you can select one connected via a PC RS232 link. If you want, the machine, and even the data-

. . . "metadata," literally, is "data about data" . . .

base of interest (there may be many databases on a single database machine), can be specified in a startup file on the PC, so the user need not even know how to connect to the database machine.

The fact that you're dealing with a relational database itself, provides the best indication of ease of use. One attribute of a true relational database is that all the information about the structure of the database *is itself stored in the database*. This information, called "metadata," literally, is "data about data," and is stored in special parts of the database called the "system relations." All databases have the same system relations, which describe things such as the data relations found in the database, the type and size (number of bytes) of all the fields in the database, the fields found in each relation, the keys or indexes in the various relations, etc. For a product such as *PC/SQL*, the system relations are a gold mine. *PC/SQL* downloads the system relations the first time it opens the database; then, it can tell the user all about the database.

How does it download the system relations? Since system relations are exactly like any other data in the database, *PC/SQL* just generates some SQL statements to query the database machine, which politely returns the system tables as data to the PC!

For example, suppose a novice user asks *PC/SQL* for information from a particular database. The user may not even know the names of the fields in the database, or the names of the relations. He doesn't have to: *PC/SQL* finds the information by querying the system relations. If the user says, "I want some information," *PC/SQL* puts up a menu with the names of all the relations in the database, and asks the user to select the one he wants, with the cursor.

Besides making navigation of the database simpler, this approach also reduces keystrokes. If there's a database you access repeatedly, *PC/SQL* allows you to save the metadata definitions on the PC. Hence, it doesn't have to perform that metadata query and download each time you do an information retrieval.

Now, there's nothing magic about the BLI database machine that makes the operation of *PC/SQL* better than with any other relational database. All "true" relational databases have the same kind of system relations, with the same kind of available information. But a tool like *PC/SQL*, lets casual users with PCs get to a vast corporate database stored on a database machine, and do useful things with the data. Since the SQL statements generated by the menus are plainly visible on the machine, *PC/SQL* also is a teaching tool to help new or occasional users learn SQL for direct database access.

There are other user interfaces for access to the BLI database machines, but as of this writing, they aren't yet fully available. Among them, is a product called *BL Today*, from BBJ Computer Services, Inc., an Australian company with offices in Santa Clara, California. This is a powerful application-generation tool closer to systems like Cognos' *Powerhouse* (see article on page 80), *Focus*, from Information Builders, Inc. of New York City, or DEC's *Rally*. Look for reviews of *BL Today* and *Focus* in upcoming issues.)

*BL Today* is written in C. It is the intent of BLI to port it to all of its major markets, including VMS, all of the UNIX variants, and PC-DOS/MS-DOS. The PC-DOS version is about to enter alpha testing, thus, it may not be too far off. Already, there are entire applications developed in *BL Today* that are on the market as standalone vertical applications running on BLI database machines. Some cost as much as $100,000.

## Distribution And Redundancy

If you spend much time looking at database solutions, you'll soon find each vendor claiming to have a "relational database." Sadly, most vendors offer products having woefully few relational features, much less adhering to the relational approach entirely. Some claim "this" or "that" relational feature, then discount the other tenets of the relational model with statements such as, "That feature's not needed in the real world," or, "That feature isn't consistent with good performance."

Today, having a relational database is not like being pregnant — you *can* be partially relational, but the most worthy vendors admit their products don't have all the relational attributes that a perfect relational database should have. They state openly that they're seeking ways to add additional relational capability. For those looking at BLI as a relational database solution, I'm happy to report that the company falls into this latter category.

## Distributed Processing

Distributed capability is one of those features that the less sophisticated vendor calls "unnecessary." But nowadays, it's not enough to have a fantastic relational solution; it's essential that it be distributed, too. Distributed processing requires more than just spreading the disks out over the computer room or the state, however; it necessitates a far more rigorous design and implementation of the relational approach. And, like all other relational features, there are various degrees of support for the distribution of databases.

If you'd like to have a database exist simultaneously at several locations, BLI database machines may be the solution. For these machines, the situation is meaningful only when the synchronization between databases is on the order of minutes, and not instantaneous. For example, one BLI customer has phone directories throughout West Germany. Each machine has many operators performing simultaneous queries and updates to the database propagate, through all the machines, and in a matter of thirty minutes or so.

This sort of update process, where copies of the database are maintained in a loosely coupled fashion, obviously would not suffice if the data were bank balances instead of phone numbers. First, bank balances change a lot more frequently than phone numbers (and mostly in a downward direction, it seems). Second, I wouldn't want to keep my money at a financial institution that allows a husband and wife to simultaneously withdraw their entire balance from two different branches!

The truly distributed database requires a two-phase commit. A "commit" is database lingo for telling the database to make the change permanent. The opposite is a "rollback," in which the transaction actually is rolled back, as if it never occurred. For you database types, a two-phase commit reduces to roughly the following steps:

1. Tell database A to prepare to commit.
2. Tell database B to prepare to commit.
3. Wait for both A and B to tell you everything is OK.
4. If both databases answer in the affirmative, tell both databases to commit; otherwise, tell them both to rollback.

A database may tell you it can't commit for a variety of reasons. It may be that someone just pulled the plug on the disk, for example. Or, perhaps it doesn't tell you anything at all — the communications line was cut by the phone company. In any case, at least you have a fighting chance at keeping both databases in synch.

I say "fighting chance" because of one small problem that I don't believe commercially available database management systems have solved: What if both databases answer the affirmative to step three, but in step four, it turns out one of them lied? For example, in that fraction of a second between the time the database says "OK," and the mediating host gives the go-ahead to commit, the communications line might go down or the disk head finally hit that proverbial floating smoke particle.

In most cases, a little human intervention, should the database say "I'm confused," isn't such an intolerable thing . . .

Simply put, although the database *claimed* it was ready to commit, it couldn't make good on the promise. This problem requires what is called a "three-way handshake," and a great deal of code to implement the solution, which only recently has been understood in the technical literature. Of course, you may not care about this relatively low-probability event, unless you are a bank making transactions for millions of dollars. In most cases, a little human intervention, should the database say "I'm confused," isn't such an intolerable thing, as long as both databases know that things are "not right" and it's something that can be fixed once both machines come back up.

BLI says it is working on a two-phase commit, but as of today, BLI database machines can't give you this Nth degree of functionality. What they *can* give is good redundancy for access to a single database, much like the shared disk access on VAXs. In this scheme, called a dual system, database machine A is given both read and write access to the database, while machine B has read-only access. This provides a good measure of redundancy if either database machine fails. Crossover from one machine to the other in the case of a failure is mediated by the host (no simple task in itself). And, since this scheme is based on dual-ported disks, what we really are talking about is redundant data access rather than true distribution of the database. Together with disk shadowing, however, this level of redundancy may be all you need in your database product.

## Is It Good Enough?

Now that we know that the BLI answer has several advantages over some of the software-only solutions available in the relational database market, you may want to know about some of the shortcomings of this approach and its specific implementation.

In Part 1, we discussed the potential pitfalls in the price/performance game. Whether price/performance is a big enough win depends on your specific circumstances with respect to capacity, expansion, the size of your existing and potential new database applications, frequency of use of the database machine, and related factors.

One potential difficulty in the BLI implementation is that it doesn't provide support for "cursors." Simply put, cursors are pointers into streams of records being passed from the database to the application. They allow the application to get a single record at a time. With multiple cursors, the application can even hang onto a record from "here," another from "there," and make decisions based on both records.

One use of cursors is to imbed an update in a query. Your program can construct a query with an update inside the loop that's retrieving the records in a one-at-a-time fashion. Then, if you decide you want to update the "current" record, a cursor will let you tell the database which record you're referencing. In the BLI case, you'll have to use the current record to construct a unique key, then submit a new record stream (from within the query record stream, if you like) to update that record. A bit of a pain, but not disastrous unless this type of operation is a frequent one in your application environment.

Another limitation is that only C currently is supported by a precompiler. Although a real database jock may want to diddle with calls to the database, for the most part you want the development assist of a good precompiler. Not only is it easier to imbed a fourth-generation language (e.g., SQL) code inside your program, it takes fewer lines of code to write an application using a precompiler. And, a good precompiler often actually can improve performance over what a non-jock might generate.

Not many data management programmers are fluent in C, and C is not known for lending itself to highly maintainable code. A FORTRAN precompiler is on its way, I'm told, but there is no mention yet of a precompiler for COBOL or other languages.

There also has been a lot of press in the past year or so about Britton Lee having difficulty keeping customers. When I asked about this, the firm admitted some problems leading to some customers abandoning their machines. It also accepts blame for some problems in the area of hardware and software support after the sale. But in at least some cases, it appears the problems were the same as for many software-only databases — the customer underestimated the resources required to get a large project up and running. Of course, with a software product, no one can tell whether it's running or not, so perhaps we shouldn't be so hard on BLI or the customers who've pulled their machines.

My assessment of the BLI offerings is that they are well crafted. There seems to be a solid commitment toward a first-rate relational database, and it certainly isn't a here-today, gone-tomorrow organization; not with roughly $30 million in annual revenue and an installed base of 600 IDMs and a bunch of RSs.

What I saw in two days (not much time), was that the firm has a good product that fills an important niche in the information management market: that of a data sharing device between dissimilar machines. I am most impressed with the richness of the user software offerings, and with the breadth of choices in hardware and host environments. There simply are not that many alternatives available if you want to access a database from PCs, VAXs, and other large hosts. The languages offered are good implementations of the various flavors of SQL, and I found the error messages to be clear and well formatted.

If you find you need a database machine, Britton Lee should be near the top of your list.

*Philip A. Naecker is a Southern California-based consulting software engineer.*

**Britton Lee Intelligent Database Machines**
Britton Lee, Inc.
14600 Winchester Boulevard
Los Gatos, CA 95030
(408) 378-7000
Price: From $50,000 (Relational Series 310) to $300,000 (IDM 500XLE).

**BL Today**
BBJ Computer Services, Inc.
2946 Scott Boulevard
Santa Clara, CA 95054
(408) 727-4464
Hardware Environment: Runs on more than 60 different operating systems and computers, including VAX/VMS, most UNIX and XENIX minis and micros, MS-DOS, and HP3000, 9000, and VECTRA systems.
Price: From $950 to $15,000, depending on configuration. Runtime library — from $150 to $6,000.

**Freeform**
Dimension Software Systems, Inc.
605 East Safari, Suite C3
Grand Prairie, TX 75050
(214) 262-8201
Hardware Environment: Britton Lee database machines; VAX UNIX, ULTRIX, VMS systems. Also, ATT UNIX System 5, IBM PC MS-DOS, Alpha Micro.
Price: From $7500 to $25,000, depending on configuration.

**OMNIBASE**
Signal Technology, Inc.
5951 Encina Road
Goleta, CA 93117
(805) 683-3771
(800) 235-5787
Hardware Environment: Britton Lee IDM or RS 310.
Price: From $21,000 for a MICROVAX II/IDM configuration, to $50,000 for a VAX 8600/IDM configuration.

**PC SQL**
Micro Decisionware
2995 Wilderness Place
Boulder, CO 80301
(303) 443-2706
Hardware Environment: Britton Lee database machines; IBM MVS, VM; Teradata database machines.
Price: $5,000 for the first five PC licenses, to $295 for 500 PCs or over.

Britton Lee, Inc.
14600 Winchester Boulevard
Los Gatos, California 95030
(408) 378-7000
Telex: 172-585