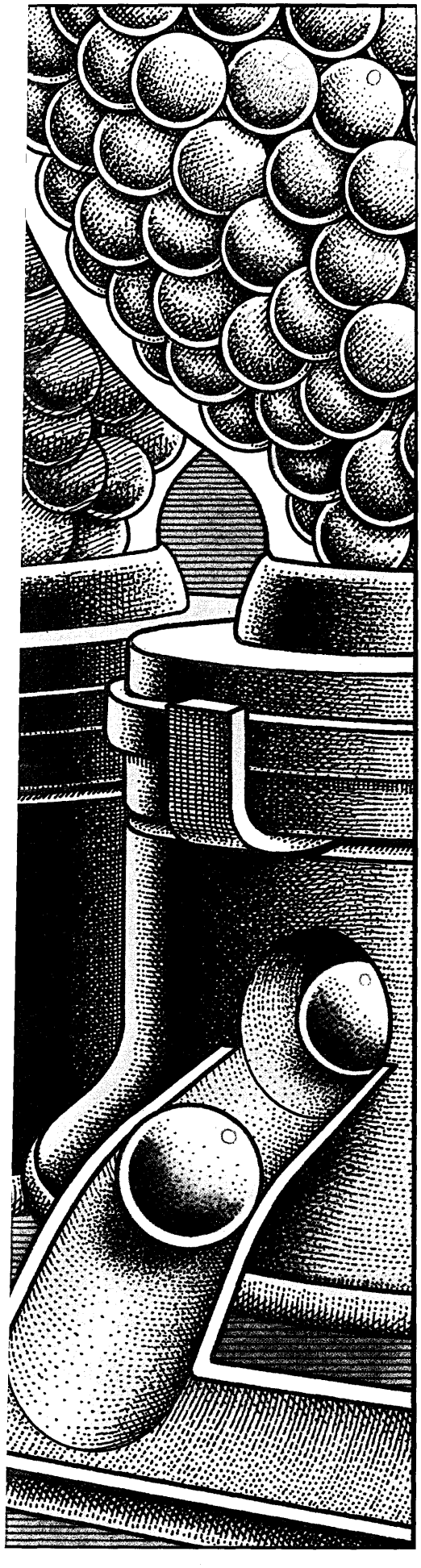

Users don't care how data are stored or sorted. They just want to choose a flavor.

DATABASE SYSTEMS FOR LOCAL NETS

by Eugene Lowenthal

A variety of forces have moved distributed data processing (ddp) from wishful thinking to reality. Among these are advances in the communication and human factors technologies coupled with rapid decreases in the cost of computing power and storage. The latent territorial instincts of the consumers of information, who want to own data and control what happens to it, have served as catalysts. Consumers have come to require a measure of independence from a central data processing (cdp) facility.

But ddp is not the nimble mammal to cdp's lumbering dinosaur, as some early visionaries proclaimed. The market for large mainframes is growing. Increasingly, ddp and cdp are viewed as solutions to somewhat different problems, and in most large enterprises there is an alliance of the two.

If it is true that ddp and cdp facilities play different roles or that they are optimized for different applications, then it is likely that their constituent building blocks will evolve differently. For instance, we would expect to see different directions taken with respect to editing languages, the styles of operator interfaces, and the attention to aesthetics in mechanical design.

Database management systems (DBMS) will also have to be different from traditional dp-oriented products if they are to be responsive to the special requirements of the distributed environment. This will be most evident in the context of the distribution vehicle known as the local area network (LAN). Important changes are needed both in the service provided by a DBMS in a LAN and its implementation, i.e., the way the service is packaged and delivered.

For purposes of this discussion it is sufficient to define a LAN as an interconnection medium characterized by:

- maximum node distances measured in meters rather than in kilometers or centimeters.
- bandwidths on the order of 1 to 20 megabits a second, and
- global addressability such that one node can potentially communicate with any other node at any time.

A well-known example of the LAN approach is the Ethernet, originated by Xerox and developed further in collaboration with Intel and DEC. The Ethernet envisions a federation of microcomputer- or minicomputer-driven elements communicating with one another through a coaxial cable. No node is in control of the network and there is no notion of master or slave implicit in the network architecture itself. Systems built on LANs, however, (including those that are Ethernet-based) will tend to be dominated by two general classes of nodes: workstations, which are operated by people and from which requests for access to global resources originated, and servers, which directly control the various global resources on behalf of all the network users. The servers field the requests and satisfy them.

In most instances a workstation is designed around a terminal (human interface) with a display and keyboard stylized for a particular application such as word processing. In the spirit of distributed processing, the workstation has some processing capability—say, for screen formatting and editing—and may have some complement of small peripherals including a floppy disk or character printer.

The central feature of a server is that it is a system resource that cannot be dedicated to a single workstation. The reason for sharing the resource may be simply economic. For instance, given the price and performance characteristics of a large laser printer, it makes sense to attach it to the network rather than make it the private property of a workstation. The big printer would be managed by a print server that has sufficient intel-

Because it is a DBMS in a box, the database server is likely to be confused with other types of database machines.

ligence to schedule the device among contending users.

Another reason for sharing a resource may be the need to make common information accessible from multiple workstations. Thus a file server serves the twofold purpose of sharing a relatively large, expensive disk and also the information that is stored on the disk. Small, private files might be maintained on a workstation's flexible disk, but large or public files would be owned by the file server. A file server must at least coordinate concurrent access to a given file from multiple requestors; a sophisticated facility would support such additional services as file sorting, catalog management, archiving, and index searching.

A database server is a file server that has gone to college. It handles not only files but databases (i.e., collections of interrelated files). It is a network node that contains a multi-user DBMS. As such, the database server represents a substantial investment in complex software and hardware. But once in place, its services and the databases themselves may be shared by all of the workstations and possibly other servers. For instance, large print files might be spooled to a database server and requested by the print server when it is ready for the job.

SPECIAL NEEDS OF SERVERS

The special requirements of database servers clearly differentiate them from other classes of DBMS products. Later on we'll focus on the contrast with conventional database systems, but some differences are worth emphasizing at the outset. In the typical DBMS environment, a single software package running on a single computer comprises the complete facility, with the end user or programmer interfacing at the top and the storage interfacing at the bottom. In the LAN environment, however, such functions as language processing and output formatting are the domain of the workstations, and they may differ greatly depending upon the application. A word processing terminal, data entry terminal, and BASIC terminal all need access to files, but each receives requests and formats output in a unique way. The objective of the database server is to provide as much of the DBMS function as possible without including the human interface or limiting what that interface can be. It must be a general purpose DBMS engine that deals with the semantics of database management but leaves the syntax to workstations. And, unlike a software DBMS, the database server must also handle conventional files and access methods so that there is no need to have both a file server and a database server (with their own sets of expensive disks) on the same network.

Because it is a DBMS in a box, the database server is likely to be confused with other types of database machines that have received a lot of attention in the past few years. But the database server is very different from the large, architecturally radical machines that have been proposed. Both rely on distributed intelligence, based on the separation of the human interface from the DBMS engine. But the design of the large database machine is motivated almost entirely by considerations of price and performance relative to mainframe software DBMS. The result may be a back end that offloads the mainframe, providing the same database bandwidth at lower cost. Or perhaps the product employs set-associative hardware to achieve transaction rates that even the largest mainframe cannot approach. Accordingly, these products are viewed as major subsystems in a slave-to-master relationship with a small number of similar large hosts to which they are connected through high-speed parallel channels.

With respect to a database server in a LAN, cost is definitely a constraining factor, performance requirements are modest, and hardware efficiency is not a burning issue. The emphasis in design is on achieving low cost for function rather than accelerating transaction rates. Furthermore, the database server is a somewhat autonomous node in a nonhomogeneous network rather than a peripheral attached to a host's I/O channel. As such, the interface to the user processes requires a complex communications protocol (instead of a specialized channel or bus protocol) to accommodate more flexible node-to-node relationships, assure reliable delivery of packets, allocate a single serial line among multiple conversations, and so on.

The overhead associated with this loose coupling, together with the relatively low bandwidth of the network medium itself (compared to an I/O channel), mandates that the interface to the database server be as con-

The integration of word processing and data processing, is, like salvation, an objective many wish to attain without quite knowing how to get there.

cise and at as high a level as possible in order to minimize network traffic. This principle was clearly understood by the designers of the Datacomputer, a large experimental database server built for a large geographically distributed network—the ARPANET. The need for a high-level database interface is also obvious in the LAN environment, and militates strongly in favor of using set-oriented data manipulation languages instead of the more primitive navigational (record-at-a-time) languages. In particular, the language proposed by the CODASYL Database task group would be inappropriate as a database server interface, because it would result in more intense interaction between the server and the workstations than is necessary given alternative higher-level approaches.

USE OF MULTIPLE SERVERS

Does it make sense to attach more than one database server to a single LAN? If the objective is simply to add capacity or increase reliability, this might be accomplished by putting two or more database servers close together so that the same disks can be shared or switched among them.

But is it ever advantageous to have truly independent database servers on the same LAN? The need is not clear unless it is to take provincialism to the bitter end: I have my database server and you have yours. In contrast to a long distance network there is no cost or response-time advantage, since within a LAN it costs just as much and takes just as long to access one node as any other. If the requirement for multiple servers is not obvious, then it is even less clear that these servers have to communicate with each other to implement a distributed database. As far as intra-LAN distributed databases are concerned, the only clear signal from the marketplace is that it is desirable to support transport of file or database subsets between the database servers and the floppies on the workstations. For instance, a page of a document can be edited at the workstation to minimize network delays and traffic.

The situation gets more interesting when the LAN includes a gateway. The gateway is a special communications server that allows nodes in the LAN to communicate with computers or terminals that are external to the LAN. For instance, the gateway may provide a link to one or more mainframes or even to the gateways of other LANs. The responsibility of the gateway is to manage these external links and translate messages between the LAN protocol and the protocols native to the remote hosts or networks.

One can envision a large central processing complex with connections to several local area networks in various departments or buildings. The cdp facility would be the residence of the corporate database, perhaps as it evolved well before the intrusion of ddp. At least two features will have to be provided in the next few years: the ability of a LAN workstation to access central files, and the ability of a database server to request subsets of central files for local retrieval, updating, and finally remerging. Similarly, we can anticipate a need to stage database subsets among communicating LANs, although today there is less market data to support this prediction.

Database subset staging is an aspect of distributed database management that is no longer technologically interesting. Most of the research today focuses on what are sometimes called transparently distributed databases, whereby a user at a workstation issues a query and the system is capable of going to any node that has data relevant to satisfying the query and formulating an answer for the user as if all the data were always at his node. The technical problems associated with implementing such a system are very complex, particularly when updating is taken into account. Moreover, the preponderance of users view transparency as blue-sky technology that will be nice to have when it works and when it's cheap. The practical requirements for the near term will be satisfied by simpler approaches. For this reason, a widely accepted commercial solution for transparently distributed databases shouldn't be expected any time soon.

All of the commercially successful database systems developed in the '60s and '70s were designed to satisfy the needs of the cdp function. The question arises as to whether new products for small machines should be clones of these earlier products. The natural response seems to be that only a subset is required. In fact the opposite is true: the DBMS for a small system must provide more services than the current generation of mainframe DBMS.

A more profound step forward is taken when the server is programmed to do something more than simple storage and retrieval for nondatabase files. For example, consider a feature whereby a file can be described to the database server as containing a document in some standard representation. Suppose further that the server is endowed with the ability to search document files in response to commands such as FIND ALL PARAGRAPHS IN DOCUMENT X THAT CONTAIN THE PHRASE ATOMIC ENERGY.

A database server that can also manipulate text becomes a far more effective building block for integrated office systems. Such a departure from strict database management is appropriate since information in the office is much richer and less structured than it is in the dp environment. For most of their history, computers have been required to deal with user data almost exclusively as formatted *records*, starting with punched cards and ending with databases. Certainly much of the information in the office can be represented as structured records, but even more can only be expressed as *streams* of text, digitized images, graphics, voice encodings, and other unbounded forms.

ONE BOX FOR TWO JOBS

The integration of word processing and data processing is, like salvation, an objective many wish to attain without quite knowing how to get there. The only clear-cut commandment at this point is that the same physical devices should be used for both kinds of jobs. Thus, the wastefulness of having a word processing terminal next to a BASIC terminal, for example, would be eliminated with new products capable of supporting both functions as needed. Likewise, the development of a database server that is able to handle both words (or pictures) and data is desirable for the same reason—minimum hardware redundancy.

The design objective of the Intel Database Processor (iDBP) was to achieve something considerably more ambitious than just getting double duty from the same tool. A key feature is the ability of the user to establish relationships between stream-oriented files and record-oriented files. Both kinds of information can coexist in the same database. In other words, the notion of database has been enlarged to accommodate new classes of data and new ways of combining data.

The iDBP is a computer that provides file and database management services to other machines (hosts) that are connected to it. The product can be configured as a back end or as a database server. In neither case is it a standalone facility; it can communicate with machines but not people.

The iDBP is positioned as a high-level building block to be incorporated as a subsystem within a complete system. It is marketed on an oem basis, primarily to manufacturers, system integrators, and software houses. It is intended for use in small multi-user systems and networks rather than in large mainframe or personal computer environments. Within these limits, any machine capable of communicating with iDBP is potentially a host. The host hardware, firmware, and software that implement the bridge between the end user and the iDBP is the customer's responsibility. The human interface thus can be tailored to the needs of the application.

The heart of the iDBP is an 8MHz 8086 microprocessor associated with up to 1 megabyte of ECC-protected random access memory. The system will accommodate up to four disk controllers, each containing its own 8089 microprocessor, and each capable of controlling up to four hard disks for a system

A strong dose of human engineering must be applied to database systems in order to make them safe for ddp.

The argument for equality has become something of a litany for computing in the '80s. The cost of hardware has plummeted to the extent that today's small systems already have more computational power, central memory, and auxiliary storage than the mainframes that originally executed today's operating systems, languages, DBMS, etc. As a consequence of Parkinson's Law, users are demanding commensurately more utility from their small systems, and are developing larger, more complex applications and databases. At the same time, the cost of application development continues to go up, particularly as a proportion of total dp costs. Therefore, programmer productivity is a vital concern that sparks the demand for ever more powerful tools and prepackaged building blocks. Personal computers aside, there really is no justification for designing only half of a DBMS.

BEYOND THE DBMS

In what ways must the DBMS for a small system go beyond conventional DBMS? It's frequently noted that the user interfaces for today's systems require the expertise of data processing professionals. Even the use of query languages requires considerable technical skill. Such demands upon the user become unattractive in the ddp environment, and totally unacceptable in the office. A strong dose of human engineering must be applied to database systems in order to make them safe for ddp.

Nonetheless, this conclusion is misleading in that it ignores a more important requirement: in order to make database systems safe for ddp, they must disappear—at least as far as the end user is concerned. Professional programmers in the ddp environment will remain comfortable with the concept of database, and they are satisfied if the vendors provide good tools and solid incremental advances in the state of the art. But database management must be invisible to end users. They will interact with their workstations to do word processing, data entry, electronic mail, or electronic filing. The DBMS that supports these functions will be absolutely vital, but hidden from view. The ultimate consumers of information will take advantage of database systems without being aware of them.

The requisite division of labor in a LAN now becomes clear. The objective for nonprogrammer workstation design is to provide the most user-friendly command language and the most palatable presentation of information. The objective for database server design is to provide a functionally rich DBMS engine. But it also must be versatile enough to successfully support the diverse information storage and retrieval needs of a variety of workstations. It is in this respect that a database server must surpass the capabilities of a traditional dp-oriented DBMS.

For instance, it was pointed out earlier that a database server is expected to manage ordinary files as well as structured databases. Thus the server can handle nondatabase objects such as program libraries, spooled printer output, message store-and-forward queues, and BASIC work files. The server need not be sensitive to the content of such files; its responsibility ends with providing logical file storage and retrieval services. File management, then, does not necessarily represent a technical achievement so much as a simple recognition that a database server has more to worry about than databases.
maximum of 16 spindles. An optional tape drive controller is also available. Rounding out the electronics is an 8086-based communications controller capable of driving up to 16 serial lines. Several packaging alternatives are available, ranging from a kit of printed circuit boards to a completely assembled subsystem in a desk-high cabinet suitable for the office environment.

The host to iDBP communications protocol is sufficiently rich to accommodate:

- The use of a single physical line for independent conversations between iDBP and multiple applications in the host.
- The use of multiple lines when iDBP backends multiple hosts.
- The deployment of iDBP as a database server in a remote network or an Ethernet-like local network.
- Extensive error detection and correction so that, for example, a telephone line can be the communications medium.

The messages that a host sends to an iDBP are actually encoded sequences of commands for defining, manipulating, and administering data. At this level, the iDBP can be regarded as a functionally comprehensive relational DBMS, supporting the definition of relations, through an active integrated data dictionary, manipulation through extremely powerful set-oriented operators, and their control through a variety of integrity, security, and recovery mechanisms. The use of high-level set operators is essential to minimizing the communication between a host and an iDBP. Traffic is further reduced through the use of a macro facility that permits any frequently used sequences of commands to be cataloged as a unit within the iDBP.

While lacking a human interface, the iDBP database facility otherwise compares favorably with commercial DBMS software products designed for mainframes and large minis. This characterization applies to internal sophistication as well. For example, database sharing by multiple on-line users is provided through a multitasking executive that interleaves command execution, a file system that coordinates concurrent access to stored data, and a recovery system that guarantees the application of interdependent sequences of updates on an all-or-nothing basis.

New DBMS products for local area networks will be very different from their predecessors.

CONNECT AND JOIN

Where iDBP departs noticeably from conventional DBMS is in its ability to handle files of arbitrary structure, including streams of text, voice/image encodings, etc. Complementing this extension to the data model are a very flexible pattern-search facility and a means for relating such files within databases. Specifically, we have coined a new relational operator called Connect. Like the relational Join function, Connect defines an associative link between two files. But where Join relates records of one file to records of another file, Connect relates records to substrings within a stream-oriented file. The substrings can be retrieved and manipulated as if they were arbitrarily long fields in the formatted record.

The ability to relate records to text (or other stream-oriented data) greatly simplifies the design of a wide variety of applications. Consider, for example:

- The use of records to represent the modular structure of a manuscript in a word processing system, such that paragraphs (and even figures) are related, sequenced, and updated through the database facility.
- The unified management of both key word indexes and text in a document retrieval system.
- The ability to merge graphics information with records to facilitate CAD/CAM applications.
- The merging of text and database data to produce reports and form letters.
- The opportunities for enhancing electronic mail when individual messages (including voice or facsimile segments) are integrated into an organized database.

Elaborating on the last example, iDBP practically forces the system designer to think about electronic mail in new ways, because the solution is perceived in database terms. For example, it is easy to see how the end product could allow the user to search for past memos written by a particular author and pertaining to a certain subject. Providing a way for the user to add marginal notations also becomes evident. Issues such as security, integrity, and recovery, which are just as important in the office as they are in the data processing shop, are already taken care of.

New DBMS products for local area networks will be very different from their predecessors, both in the manner of implementation and the way they are applied. They will play a key role in unifying data processing and office functions by providing a holistic approach to managing and sharing all classes of information.

As integrated office systems become more sophisticated, the need for powerful underlying database tools will become increasingly obvious. But while database technology as such is approaching maturity, there is little understanding today of how it must adapt to the emerging office environment. As we learn the role of database servers and back ends as building blocks for the electronic office, that should change. ✲

In 1970 Eugene Lowenthal joined MRI Systems Corp., where he was chief architect of the System 2000 DBMS and ultimately became vice president of advanced development. After Intel Corp. acquired MRI in 1979, he initiated the database processor program, which he continues to direct. He has a bachelor's degree from the University of Chicago and a doctorate in computer science from the University of Texas.