Transaction processing systems on future workstations: A feasibility study

by JACOB SLONIM, JOHN HENSHAW and AVI SCHONBACH
Geac Computers International
Ontario, Canada
and
MICHAEL BAUER
The University of Western Ontario
Ontario, Canada

ABSTRACT

An account of a benchmark test to evaluate the performance of a relational database management system, INGRES, in the context of a library circulation system. The results suggest that, within a couple of years, relational database systems running on microcomputers within distributed environments will be performance- and cost-effective in supporting transaction processing systems.
INTRODUCTION

Organizations such as banks, airlines, and libraries rely heavily on transaction processing systems in their day-to-day operations. These systems provide rapid on-line access to information for both customers and employees. The emergence of powerful microcomputers and reliable communication has created an opportunity to develop distributed systems in which data locality reflects the locality of users.

The term "workstation" is used to represent a comprehensive set of tools tailored to a specific type of user—in Geac's case, a librarian's workstation. The functions of these tools are independent of the machine size. The purpose of this research is to define a minimum standard configuration for hardware and system software to cost-effectively support the workstation toolset.

We at Geac are currently involved in a project which calls for the development of a commercial transaction processing (TP) system in a distributed environment for use in libraries. Our goal is a TP system capable of operating in a large distributed environment composed of microcomputers, minis, and mainframes. This system, moreover, is to be based on existing, off-the-shelf hardware and software. These goals have led us to consider several relational database systems as a basis for the TP system, and to consider UNIX as the basic operating system.

Historically, commercial transaction processing systems have relied on specialized databases, such as Tandem's ENCOMPASS/TFM or Geac's Geacos, or on traditional hierarchical and network databases such as IBM's IMS/TPF. These systems are commercially available, and a great deal is known about their behavior and performance. In contrast, little is known about the use of relational database systems in TP environments.

Although relational database management systems offer a number of advantages in distributed transaction processing (discussed in detail below), a common argument against their use has to do with performance. Two widely held opinions can be summarized as follows:

Relational systems are all very fine for ad hoc query, but they will never achieve the performance needed for production systems or transaction processing systems...

Relational systems require a breakthrough in hardware technology (e.g., hardware associative memory) before they will be able to achieve acceptable performance.

In contrast, the opinion of the authors and many other researchers is that:

there is no intrinsic reason why a relational database system should have worse performance than traditional database systems.

The next two sections discuss UNIX and the advantages of relational database systems. Thereafter follows a description of our benchmark of a commercial relational database management system in a transaction application.

WHY THE UNIX OPERATING SYSTEM?

As we have noted, one of the original aims of the project was to rely as much as possible on existing commercial hardware and software, and to take advantage of new technology as it appears on the market. This aim, coupled with requirements to support distributed systems with powerful microcomputers and reduce the cost of developing new software, led us to adopt AT&T UNIX as a logical choice for our operating system.

In practice, the development of applications to operate on multiple operating systems is a long, expensive process, and one which often results in software of poor quality. Servicing user needs across hardware from various vendors and of various sizes becomes much easier when a single operating system is used. A single operating system greatly simplifies the incorporation of industry standards in networking, database management, and other software tools.

Why UNIX rather than some other operating system? We have chosen UNIX because its design stresses backward compatibility with previous versions of itself, and thereby protects the investments of Geac and its customers. This feature allows the Geac family of computers to communicate with a growing installed base of UNIX applications within the industry. The flexibility of this operating system allows Geac to add enhancements to our existing system without compromising the hardware or software investment of our customers. Since UNIX is offered by an increasing number of vendors and used at an increasing number of universities, the rate of product innovation will continue to grow rapidly.

All UNIX operating systems are not alike, but efforts are under way to find unity in this diversity. One approach is that adopted by AT&T, which has issued the Full System V Interface Definition, Issue 2, as well as a validation suite. The IEEE's approach is more comprehensive: the P1003 Standard Committee is working on a single UNIX operating system standard for worldwide use.

UNIX is by no means without its difficulties. For example, UNIX's i-node structure poses performance problems in the handling of large databases, and a single file cannot span multiple spindles. There is also a general concern about known bugs and built-in overheads.
WHY A RELATIONAL DATABASE MANAGEMENT SYSTEM?

A database management system is a large, complex collection of software routines positioned between the user’s application program and the data to be processed. The DBMS controls access to and manipulation of the data on behalf of the application programs. Data models, which organize data logically according to genuine relationships in the data files, have been developed out of either graph or set theory. The three primary models in use today are the hierarchical, network, and relational.

All three have advantages and drawbacks. The network and hierarchical models have reached commercial maturity, while the relational model has received a great deal of commercial attention in recent years. Unlike the traditional models, which have been used in the computer industry for years, the relational model has evolved along with the microcomputer. From 1983 through 1985, the number of commercially available relational database systems increased from 40 to more than 100. It is clear now, with products like DB II/III and ORACLE on microcomputers, and with companies like IBM announcing DB2 (System R) and Cullinet announcing IDM/R, that many vendors feel that relational databases represent a viable commercial technology.

The relational database model traces its roots to theories developed in relational mathematics. The artificial set constructs, intrinsic to the hierarchical and network schemes, are not relevant to the relational model. Instead, data relationships are reduced to simple components and represented directly through views of data relationships. The database itself is homogeneous, and this homogeneity makes it possible to define any number of data relationships or logical views of data, and to process it by performing logical operations on attributes.

The relational model is a way of looking at data—a prescription for the representation and manipulation of data. This prescription has three components:

1. **structure**, that is, tables (rows and columns of data)
2. **integrity**, that is, a means of ensuring, for example, that every relation (table) has a unique key to identify table entries or rows, and
3. **manipulation**, consisting of operators for processing tables; these operators are straightforward: the *select* operator picks out rows; the *project* operator picks out columns; the *join* operator combines two tables.

These characteristics of the relational model provide real advantages in a distributed transaction processing environment. The relational model generally shields the application designer from the complexity of storage structures, data definitions and the design of access paths. The labor costs associated with database implementation and maintenance are thereby lowered.

Today, relatively high performance is offered in database management systems founded on the traditional (hierarchical and network) models. Nevertheless, the traditional models do have their drawbacks. The most notable is that they require a high level of effort on the part of the application designer; invariably the application designer must specify complex storage structures, data access paths, and data definitions. The network model is also inflexible, in that access paths that are not predefined when the database is loaded cannot be introduced without major restructuring of the database.

One criticism of relational systems is that they are primarily aimed at supporting query requirements, and consequently are not well suited for full-scale production and/or transaction processing. It is true that no existing commercial relational product can perform as well as, for example, IMS Fast Path. The reason for that could well be that IMS Fast Path runs on very large machines, like the Sierra 400. There are, however, no inherent theoretical or practical reasons why relational systems cannot ultimately match the top performance of traditional database systems.

An advantage of the relational model is that it prevents the application developer from seeing explicit connections or links between tables (that is, physical pointers), and thereby avoiding traversal between tuples on the basis of such links. It also precludes user-visible indices on attributes, and removes the physical storage structures from the concern of users. The relational tables are a logical abstraction of what is physically stored.

The tables in a relational DBMS are a normalized structure. In systems like IMS and IDMS the physical structure is biased toward certain applications and machine architectures by the inclusion of built-in access paths. Consequently, for those particular applications, the network and hierarchical models can be very powerful. The relational model, on the other hand, permits the dynamic creation of access paths through the use of the manipulative operators. Since the application is no longer limited to predetermined access paths, data independence of the system is enhanced.

In relational systems, in contrast to traditional database management systems, the theory preceded any implementation. If a relational implementation conforms to this theory, its behavior in any given situation is completely predictable.

Under traditional database management systems, some actions may result in unpredictable events. For example, one application might delete a record which is linked to other records, with the result that none of these records are accessible to other applications.

A standard query language is an important issue because it renders database definitions and application programs portable among implementations conforming to the standard. Both ISO and ANSI have established committees to develop standard query languages, for both the network and relational models. The ANSI X3H2 committee is at the draft proposal stage for the network model. The same committee has already reached agreement on a standard for the relational model based upon SQL. ISO has also approved this standard.

The standard for the relational query language applies to implementations in an environment that may include application programming languages, end-user query languages, report generator systems, data dictionary systems, program library systems, and distributed communications systems, as well as various tools for database design, data administration, and performance optimization.

The SQL language was developed in 1974 at IBM. The
technology explored and developed in System R was subsequently exploited by IBM in both SQL/DS and DB2. Recently, ORACLE and Fujitsu have announced an SQL product. Last year, INGRES introduced their version of SQL in conjunction with supporting QUEL. QUEL was originally developed in 1974 by M. Stonebraker and others at the University of California at Berkeley as part of the INGRES system. Of the 100 or so relational products on the market, at least 30 have an SQL flavor.

In distributed database systems, one important requirement is that communication traffic be minimized. Relational database systems have a number of characteristics which make them an excellent choice in distributed systems. First, the set handling capabilities within the relational data manipulation language lead to a more effective use of communication lines; the system receives one request for each set of tuples required, rather than one per tuple. Under the network model, application programs operate entirely in a one record-at-a-time mode.

Second, the relational model makes it straightforward to partition tables either vertically (columns) or horizontally (rows); these partitioned tables can then be easily distributed across the network. The standard relational operators, join and union, can be used to reassemble the partitioned tables. Both these areas present considerable difficulties for systems based on traditional models.

Third, relational operators are high-level, and for that reason they can be optimized in ways essential for distributed access plans (System R).

Fourthly, the relational model provides data independence for applications. An application program is data independent if it does not require modification when the database is restructured or reorganized. Program data independence is provided by hiding from the application program the physical placement and organization of data in the database. This is particularly important for location transparency within distributed databases. In traditional systems the application must use predefined physical pointers to access the data.

An important implication for application development is that relational systems simplify prototyping. Using a relational database management system it is easy (2-3 weeks) to design and create a database, build some application (e.g., a library circulation system), and then run a prototype featuring actual screens and reports.

THE QUESTION OF PERFORMANCE

We have outlined a number of reasons why relational databases and UNIX are good tools for building distributed systems and applications. These tools will allow us to take advantage of future developments in hardware technology because of their independence from any particular hardware.

Developments in large-scale, very large-scale integration (LSI/VLSI) and integrated circuit (IC) chip technology have led to package miniaturization, minimal interconnections, economy of scale, and increased functions on an IC chip. As a result, more powerful microcomputers are continually appearing. Microcomputers are now available which provide large main memories (e.g., up to 16MB on a MicroVax II), support large mass storage devices (e.g., 420 MB drives on a MicroVax II), and provide hardware support for memory management (e.g., the INTEL 80386 and Motorola 68030). Enhanced communication capabilities, such as Ethernet, are already readily available. As hardware technology becomes more sophisticated, more software functions can be embedded within the hardware (e.g., memory management and communication protocols). This provides an opportunity to increase overall system performance, and in particular, transaction processing performance.

However, these advantages and trends do not in themselves guarantee that one can build commercially viable transaction processing systems. There remains the unresolved question of performance. To address this concern, we decided to benchmark a commercial relational database management system under UNIX in a transaction processing environment.

THE LIBRARY CIRCULATION BENCHMARK

Market requirements for our project dictate a performance of one transaction per second per $25,000. In today's market, this corresponds to the cost of a microcomputer with the performance of a Vax 8650. The purpose of our benchmark was to determine whether commercial relational database products would be capable of this level of performance on microcomputers likely to emerge within the next two years. INGRES release 4.02 was the relational database management system tested. INGRES is now marketed by Relational Technology Inc. and runs on a variety of computers. Since we were interested in the performance of "standard" commercial database systems, no tailoring of the INGRES software was made with respect to the requirements of the benchmark, in which, by the way, Relational Technology Inc. played no role.

The benchmark chosen was a library circulation system. On the basis of marketing requirements for a typical library system, a performance requirement of 5 transactions per second had been established. This is comparable to the Debit-Credit transaction benchmark in terms of the number of reads and writes per transaction. The relationship to other transaction systems is illustrated in Table I. The Debit-Credit system is used as a basis for the comparison of transaction processing systems. Table I lists several different systems, presents their requirements in transactions per second, and gives a weight based upon the Debit-Credit system.

<table>
<thead>
<tr>
<th>Application</th>
<th>Transaction Rate/Sec</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>lottery</td>
<td>400.0 tps/ cpu</td>
<td>0.01 D-C</td>
</tr>
<tr>
<td>800-number</td>
<td>50.0 tps/cpu</td>
<td>0.10 D-C</td>
</tr>
<tr>
<td>video text</td>
<td>20.0 tps/cpu</td>
<td>0.20 D-C</td>
</tr>
<tr>
<td>credit authorization</td>
<td>10.0 tps/ cpu</td>
<td>0.40 D-C</td>
</tr>
<tr>
<td>debit-credit</td>
<td>4.0 tps/ cpu</td>
<td>1.00 D-C</td>
</tr>
<tr>
<td>&quot;real&quot; debit credit</td>
<td>2.0 tps/ cpu</td>
<td>2.00 D-C</td>
</tr>
<tr>
<td>electronic mail</td>
<td>0.2 tps/ cpu</td>
<td>20.00 D-C</td>
</tr>
<tr>
<td>phone store</td>
<td>0.1 tps/ cpu</td>
<td>40.00 D-C</td>
</tr>
</tbody>
</table>
Environment

The benchmark was run on a VAX 8650 (8 MIPS), configured with 16 megabytes of memory, and using 120 megabytes of a 420 megabyte disk spindle. The host operating system was Ultrix 1.2 (BSD4.2). This configuration reflects our prediction of the performance of microcomputers available within a very few years. Benchmark runs were executed when there were no other active users.

An INGRES page is 2K bytes, representing four Ultrix virtual memory pages. During the course of the benchmark INGRES process cache size varied from 9 to 250 INGRES pages (18K to 500K bytes).

Ultrix imposed a number of limitations on the benchmark. In our configuration the available Ultrix lock table resources and virtual memory are exhausted when ten concurrent processes using the maximum page allocation per process are running. When this happens, INGRES cannot proceed and shuts down gracefully; INGRES processes that have sufficient resources continue to execute. Lock table size and available virtual memory size are system parameters; a system reconfiguration is necessary to set different values. Another constraint on the benchmark imposed by Ultrix is that a maximum of 24 processes are permitted for each login session; this maximum is also an operating system parameter. Since each INGRES invocation requires a “front end” and a “back end” process, a maximum of twelve process-pairs could be run at once. In practice, this number is actually a bit lower because of the Ultrix process required to maintain a user session.

Methodology

The benchmark library circulation system involves three types of transaction: charge a book, discharge a book, and put a hold on a book. A skeleton of the current Geac library circulation system was used in the design of the benchmark. This skeleton was implemented on facilities provided by INGRES.

The benchmark data was taken from a medium size public library in British Columbia, Canada. This data was chosen because it is used internally at Geac for quality assurance testing of the current circulation system. The original database consisted of running the same 2000 transactions to completion. The columns indicate the cache sizes available to and used by each INGRES process, the number of INGRES processes executing concurrently, the total CPU time, the elapsed time, the number of direct I/O requests (i.e., to the operating system to access the disk), cache request—the number of times the cache is accessed to read a data page, cache

Results

A summary of the benchmark results appears in Table IV. Each table entry represents one run of the benchmark, which consisted of running the same 2000 transactions to completion. The columns indicate the cache sizes available to and used by each INGRES process, the number of INGRES processes executing concurrently, the total CPU time, the elapsed time, the number of direct I/O requests (i.e., to the operating system to access the disk), cache request—the number of times the cache is accessed to read a data page, cache
read—the number of times the cache must read a page from the disk, cache write—the number of times the cache writes a page to disk, and the number of transactions per second. The number of transactions per second is the number of transactions divided by the elapsed time. The following observations can be made regarding these results:

1. The best performance achieved was 4.3 transactions per second. The worst result observed was 2.4 TPS.
2. The number of direct I/Os is of primary importance. Performance decreases with an increase in the number of direct I/Os.
3. As the size of the INGRES process cache increases, the number of direct I/Os and the number of cache reads decreases.
4. The numbers of cache requests and the number of cache writes remain relatively constant, independent of INGRES process cache size.
5. The ratio of cache reads to direct I/Os is significant. An increase in the ratio decreases elapsed time. This is, however, of secondary importance to the number of direct I/Os. An increase in the number of concurrent processes causes an increase in this ratio.
6. When the number of direct I/Os is constant and the number of concurrent processes is increased, performance improves.
7. There is a strong correlation between the number of concurrent INGRES processes and the use of the available cache per process. Too many concurrent processes, however, create overhead as a result of competition for access to the data. Best performance is achieved at the point where the addition of another concurrent process causes a less than maximum use of available cache.

Experiments varying the item and patron table structures were performed on the small database with a cache size of 23 pages per INGRES process. These two tables are modified by every transaction. Table V presents these results. The ISAM access method provided the best results for our data. The hashing based storage structure failed to provide reasonable performance, since the keys in the tables were unsuitable for the INGRES hashing algorithm. The B-tree access method, used by the current Geac circulation system, is available with INGRES 5.0. This access method could offer an additional improvement in performance.

### TABLE V—Benchmarks with varying file structures

<table>
<thead>
<tr>
<th>Item</th>
<th>Patron</th>
<th>proc</th>
<th>IO reqs</th>
<th>cache reqs</th>
<th>read</th>
<th>write</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash</td>
<td>hash</td>
<td>1</td>
<td>71783</td>
<td>145888</td>
<td>144174</td>
<td>5203</td>
<td>0.4</td>
</tr>
<tr>
<td>hash</td>
<td>hash</td>
<td>2</td>
<td>76134</td>
<td>151522</td>
<td>141423</td>
<td>5204</td>
<td>0.4</td>
</tr>
<tr>
<td>ISAM</td>
<td>hash</td>
<td>1</td>
<td>29736</td>
<td>76033</td>
<td>70307</td>
<td>4197</td>
<td>0.9</td>
</tr>
<tr>
<td>ISAM</td>
<td>hash</td>
<td>2</td>
<td>35122</td>
<td>82937</td>
<td>7133</td>
<td>4198</td>
<td>0.8</td>
</tr>
<tr>
<td>ISAM ISAM</td>
<td>1</td>
<td>10002</td>
<td>13543</td>
<td>7133</td>
<td>4034</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>ISAM ISAM</td>
<td>2</td>
<td>10471</td>
<td>13576</td>
<td>7732</td>
<td>4036</td>
<td>2.9</td>
<td></td>
</tr>
</tbody>
</table>

**SUMMARY AND CONCLUSIONS**

Relational database management systems are a viable option in commercial transaction processing. The combination of a relational database management system and the UNIX operating system offers savings in development cost, portability among vendors and computer architectures, and the ability to take advantage of future technological innovations. UNIX is a solid platform for distributed database management systems.

UNIX and relational database management systems were first implemented, and first achieved commercial success, on small computers. They are now becoming commonly available on large and very large computer systems.

The benchmark experiment demonstrated to our satisfaction that a relational database management system can provide sufficient performance to meet market demands. In order to achieve commercial success in our market area, we must reduce the cost of a TPS. This reduction can be achieved only when the performance of the VAX 8650 is made available on a small computer like the MicroVax II. Both computers use the 32-bit word size required to run a large database management system efficiently. The benchmark results indicate that the critical performance factors are the disk access time and the size of main memory. The minimum main memory size of the Vax 8650 is already available on the MicroVax II.

We limited the benchmark to a single disk spindle, in order to match as closely as possible the hardware configuration of existing microcomputers.

Since direct I/O is the most critical factor in the performance of the system, in situations in which more than one disk is available, the tables could be stored in such a way as to allow parallel I/O. In our system, for example, patron and item tables would then reside on different disks.

Having started with the hypothesis that the relational model is capable of sufficient performance, we have reached the conclusion that by using one of several commercially available relational database management systems we can build a commercially successful library transaction processing system.

**REFERENCES**

2. Information on the Geac operating system is available from Geac Computers International, 350 Steelcase Road West, Markham, Ontario, Canada, L3R-1B3.
4. Date, C.J. Relational Database—Selected Writings, Reading, Massachusetts: Addison-Wesley, 1986, p. 66.


15. IBM Corporation. SQL/Data System General Information, IBM Form no. GH24-5012.

16. IBM Corporation. IBM Database 2 General Information, IBM Form no. GC26-4082.

17. Information on INGRES is available from Relational Technology Inc., Alameda, California.

