Patricia Seybold's

UNIX IN THE OFFICE

VOL. 2, NO. 10
ISSN: 0887-3054
OCTOBER, 1987

INSIDE

PRODUCTS • TRENDS • ISSUES • ANALYSIS

The Oracle Relational DBMS

The Leader of the Pack

By Judith R. Davis

ORACLE CORPORATION TAKES a great deal of pride in the fact that its Oracle relational database management system (RDBMS), introduced in 1979, was ahead of its time. Oracle was the first commercial RDBMS, modeled on research conducted by IBM in the early 1970s, and the first commercial DBMS product to use IBM's Structured Query Language (SQL) to access data in the database. (IBM published the specifications for the SQL data language in 1976, and this became the basis for the development of Oracle.)

The second RDBMS, Ingres from *(continued on page 3)*

• E D I T O R I A L •

The Next Blockbuster Is Just around the Corner

Cooperative Software will be 1988's 1-2-3 equivalent.

By Patricia B. Seybold

WHAT SOFTWARE will fuel the next boom in computing? What will be the next blockbuster product that sends us all stampeding out into the marketplace? We'll place our bets on the first application program that truly helps groups of people work together more effectively. That program could be just around the corner. It will certainly make its debut by the end of 1988. "Teamwork and technology" is one of the major themes for the 1987 Seybold Executive Forum (November 9th through the 11th). Why? Because every major systems house and software company we talk to is working on a product to address collaborative work. The office computing industry's software designers (in Europe and in the United States) are all focusing on the same target: tools for cooperation.

Software and system designers learned the need for such tools from their bitter experiences trying to develop products and bring them to market on time, within budget, and as spec'd. Invariably, one of those goals is missed. And the process of bringing a product to market is rarely a pleasant one for all involved. Too often, chronic burnout on the part of the developers and lukewarm reception on the part of the marketplace stifles the team's enthusiasm for a rematch.

What these computer companies all know is that the existing tools that they all use—networked computers, electronic mail, electronic calendars, scheduling systems, and even project management software—don't really reduce the complexity of collaboration and teamwork. New tools are needed that will actually support and augment the collaborative process.

What will those tools do? They will streamline effective communication; support the collaborative creation of documents, specifications, research, and code; enable more efficient brainstorming; provide ways to effectively alert all team players to changes in schedule and emphasis. These tools may even assist in predictably creating synergy—that magic creative energy which envelops a group of people when they are really cooking, and which causes pettiness and protocol to drop out of the picture.

This new breed of applications—cooperative software—is not being developed solely for selfish motives on the part of computer companies. They believe that this kind of solution is universally needed by their customers. By the time the marketplace recognizes this need and the demand for cooperative software builds, these vendors hope to have deliverable products to satisfy users' new-found requirements.

The need for collaborative tools may be more acutely felt by computer firms because they already are prolific users of their own computer technologies. They know what technology will do for them and what problems it doesn't solve. Customers, on the other hand, have less firsthand experience with the technology. They have lower penetration of electronic mail, massive networks, and the like, and they naively assume that broader installation of the tools available today will bring greater payoffs in group productivity.

While IBM and DEC lock horns in mid-range computers and enterprise networking, the real industry story percolates just beneath the surface. Stay tuned for products from Lotus, Hewlett-Packard, and some as yet unknown players. We predict that once a tool or a set of tools arrives on the market with the right combination of usability, transparency, and real efficiency, it will take off with the same thrust that characterized Lotus's 1-2-3, propelling us into a new phase of office computing.

Patricia Seybold's Office Computing Group

Editor-in-Chief PATRICIA B. SEYBOLD
Managing Editor RONNI T. MARSHAK

Senior Editor
MICHAEL D. MILLIKIN
News Editor
JUDITH S. HURWITZ
Staff Writer
DAVID S. MARSHAK
Sales
RICHARD ALLSBROOK JR.
Customer Service
DEBORAH A. HAY

Associate Editors
JUDITH R. DAVIS
11 Ellery Square, Cambridge, Massachusetts 02138
Telephone: (617) 876-4081

DAVID L. TERRIE
135 Vernon Road, Scituate, Massachusetts 02066
Telephone: (617) 545-7401

Contributing Editor
GARY J. NUTT
3450 22nd Street, Boulder, Colorado 80302
Telephone: (303) 444-6341

148 State Street, Suite 612, Boston, Massachusetts 02109 Telephone: (617) 742-5200 FAX: 1-617-742-1028

•COMPANY•

(continued from page 1) Relational Technology, didn't hit the market for another two years, and the next SQL-based contender, IBM's SQL/DS, didn't appear until 1982. More and more of Oracle's competitors are jumping on the bandwagon as "relational" and "SQL-based" have become requirements for a successful DBMS in today's market.

In this, the fourth in our series of reviews of RDBMSs, we take a close look at Oracle to see if the company and its product are still maintaining a leadership position in this fast-changing industry.

Company Background

Oracle Corporation was founded in 1977 by Larry Ellison, president, and Bob Miner, senior vice president of development. The Oracle RDBMS was originally developed and introduced on a DEC PDP/11 under RSX11. That same year (1979), Oracle was rewritten in C and ported to the VAX/VMS environment. The decision to go with DEC was made for two important reasons. First, DEC already had a large number of installations, and Oracle guessed accurately that DEC would continue to be successful. (Relational Technology made that same decision two years later, when it introduced Ingres.) Secondly, at the time, IBM offered database capabilities only on its mainframes under MVS and VM, not in the minicomputer arena. Thus, going with DEC provided two important marketing platforms for Oracle. And we all know how phenomenally well DEC has done. In fact, Oracle gives some measure of credit to DEC. Bruce Cleveland, director of Unix Product Line Marketing for Oracle, states, "DEC's success launched Oracle's success."

In the past, Oracle's development and introduction of new releases have been in the VAX/VMS environment. More recently, however, the company has made a dramatic commitment to Unix. Most of Oracle's internal development is being moved to Unix, and everyone on the development staff has a Sun workstation connected via Ethernet to a Sequent back end. New products are now introduced in Unix and VAX/VMS almost simultaneously.

FINANCIALS. Oracle went public in May 1986. Revenues soared from $55 million to $131 million for the fiscal year ending in May 1987, an increase of 138 percent. And the company states it is on target to pull in $235 million this year, an impressive increase of 80 percent.

STAFFING. Oracle employs 1,600 people worldwide, up from a staff of only 200 two years ago. That is a phenomenal growth rate, and it raises the obvious question about how well the company is managing that kind of growth.

Product Line

Oracle is an SQL-based RDBMS plus a set of integrated software tools for application development and decision support. The applications development tools include forms generation and interfaces to most major programming languages. Decision support tools offer a Lotus-like spreadsheet and high-resolution color graphics support. The company unbundled its products with Version 5.1 in September, 1986. The database kernel and tools are now each licensed separately. Thus, customers can mix and match to meet their own needs.

CORE PRODUCTS. The Oracle RDBMS, the back-end database management kernel, is the core of the Oracle product line. The latest version is 5.1, which includes capabilities for distributed databases. SQL*Plus provides a command-driven interface to the RDBMS for creating and maintaining the database, and ad hoc queries and reports. The SQL*Forms module provides a forms-generation tool; the resulting custom forms can be ex-

Platforms for the Core Oracle Products

The core Oracle products (the RDBMS, SQL*Plus, SQL*Forms, Pro*C, and Pro*Fortran) all run on the following platforms:

Unix

many, including:

- Amdahl (UTS)
- Apollo (Aegis)
- AT&T 3B (System V)
- Convergent Technologies (CT/IX)
- Data General MV (DG/UX)
- DEC VAX (Ultrix)
- Harris (HCX-7)
- HP 9000 (HP-UX)
- IBM PC/RT (AIX)
- NCR (System V)
- Plexus (System V)
- Pyramid (OSX 3.1)
- Sequent (Dynix)
- Sun (SunOS 3.2)
- Xenix

Non-Unix

- Data General MV (AOS/VS)
- DEC VAX (VMS)
- Honeywell (GCOS)
- IBM mainframe (MVS, VM/CMS)
- IBM PC (DOS)
- IBM System 88 (VOX)
- Prime (Primos)
- Stratus (VOS)
- Wang VS

The Oracle Family of Products

Product	Introduced	Currently Runs On	Cost
Oracle RDBMS (Includes SQL*Report)	1979	See box on page 3	Ranges from $1,500 on PC under Xenix up to $144,000 on Amdahl
SQL*Plus (Interactive command utility for ad hoc data access and report writing)	1979	See box on page 3	20% of cost of Oracle RDBMS
SQL*Forms (Application generator and run-time system; features screen-painting for applications development and prototyping)	1979	See box on page 3	25% of cost of Oracle RDBMS
SQL*Menu (Allows creation of menu trees)	1986	DEC VAX/VMS, will be available on all Unix platforms by 10/87	20% of cost of Oracle RDBMS
SQL*Design Dictionary	1986	DEC VAX/VMS, NCR (System V), IBM PC/RT (AIX), IBM PC (DOS)	Not available
SQL*Graph (Generates pie, line, bar charts in color)	1985	DEC VAX/VMS	20% of cost of Oracle RDBMS; included in Easy*SQL
SQL*Calc (Lotus-like interface to Oracle RDBMS)	1986	DEC VAX/VMS, IBM PC (DOS), coming in Unix	20% of cost of Oracle RDBMS; included in Professional and Networkstation Oracle
Easy*SQL (Menu-driven interface to Oracle)	1985	DEC VAX/VMS, coming for PC, Unix, DG, Wang, Prime, etc. (4Q '87)	20% of cost of Oracle RDBMS
Pro*SQL (Programmatic interfaces for Cobol, C, Fortran, Pascal, Ada)	1985-1987	Varies depending on platform	15% of cost of Oracle RDBMS (each)

The Oracle Family of Products

Product	Introduced	Currently Runs On	Cost
SQL*Star architecture (Provides open-system distributed RDBMS)	1986		
—SQL*Net (Provides network-independent connection between application and database)		All platforms where Oracle RDBMS V.5.1 currently runs	20% of cost of Oracle RDBMS
—Distributed Query Facility (Provides location-independent execution of queries)		Same as above	Included in RDBMS V.5.1
—Network Protocol		Oracle currently supports DECnet, TCP/IP, asynch, and 3270	15% of cost of Oracle RDBMS (each)
—SQL*Connect (Gateway interface to non-Oracle databases)	Announced 1986; scheduled availability early 1988	Support for DB2 is in beta test; coming is SQL/DS and other SQL-based DBMSs.	Not available
Professional Oracle (Includes Oracle RDBMS V.5.1, SQL*Plus, SQL*Forms, SQL*Calc, SQL*Report, Pro*C)	1987 (First PC product was introduced in 1984)	IBM-compatible 286, 386, PS/2 model 50 up under DOS 3.x (requires 1.5MB memory)	$1,295
Networkstation Oracle (Front-end only; includes SQL*Plus, SQL*Forms, SQL*Calc, SQL*Report, SQL*Net)	1987	IBM-compatible PC (512K)	$695
LANserver Oracle (Includes Oracle RDBMS V.5.1, SQL*Plus, SQL*Net)	1987 (Will be available in 12/87)	Same as Professional Oracle but requires a minimum of 3MB memory; 1st release will support TCP/IP on Novell Advanced Netware Ethernet	$2,495

panded into full applications with what Oracle calls "triggers." Triggers define database processing associated with actions taken on a form. Both SQL*Plus and SQL*Forms have been available in one form or another since Oracle was first introduced.

DEVELOPMENT TOOLS. In addition to SQL*Forms, Oracle has two other relatively new products for applications developers. SQL*Menu lets the developer create customized menu trees for applications. SQL*Design Dictionary (SDD) is a tool for designing an application from scratch, a "fill in the form" approach to describing an application. (It was written using SQL*Forms and the RDBMS.) SDD then creates a database and produces a prototype design.

Oracle's first computer-assisted software engineering (CASE) product, SDD was developed in the United Kingdom. Oracle's international group essentially runs its own business, since selling must be done differently outside the United States. According to Cleveland, approximately 90 percent of Oracle's international sales include an applications development method-

ology and consulting in addition to software products. Oracle has developed what it calls "business patterns" using SQL*Methodology, a process for putting an application together. The international staff developed SDD to help with this process, and has subsequently built a library of business applications. Oracle can take an application out of the library and then tailor it to the customer's specifications.

Both SQL*Menu and SDD run in the DEC VAX/VMS environment; SDD also runs on a few other systems, such as the NCR Tower, that have proven important to Oracle's international business. Since the international staff is responsible for all of Oracle's CASE products, the U.S. group must wait for a porting kit to expand SDD's horizons. In general, the company plans to first port SQL*Menu to other platforms and follow it with SDD.

*The SQL*Star architecture provides two major advantages: its location transparency, and its support of joins and unions across network nodes.*

HOST LANGUAGE HOOKS. Oracle supports two levels of interface with traditional third-generation programming languages (3GLs): Precompilers (Pro*C, Pro*Cobol, etc.) allowing embedded SQL in programs written in C, Cobol, Fortran, PL/I, Pascal, and Ada; and a predefined set of subroutine calls (Pro*SQL) as a host language interface that can be used with any programming language. Pro*SQL provides lower level calls to access data in an Oracle database directly, rather than through SQL.

END-USER TOOLS. Oracle has recently developed several modules that are targeted primarily at the end user. These include Easy*SQL, a menu-driven interface for Oracle databases, and two decision support tools: SQL*Calc (a spreadsheet) and SQL*Graph.

SQL*Calc. What Oracle brings to the table with SQL*Calc is a Lotus-like spreadsheet with the ability to embed SQL statements in cells. With SQL, the user can extract data from an Oracle database and enter it into a range of cells on the spreadsheet. The SQL statement can refer to another cell on the spreadsheet, enabling the user to change the value of variables and extract a different set of data from the database. The user can also create tables and enter data into tables from the spreadsheet. Oracle is the only RDBMS vendor, to our knowledge, that provides such a tool.

Later this year, the company plans to introduce 1-2-SQL. This is an add-on to Lotus 1-2-3 that will enable the Lotus user to also embed SQL statements directly in Lotus spreadsheets and access an Oracle database.

SQL*Graph. An optional extension of SQL*Plus, SQL*Graph generates pie, line, and bar charts from data in an Oracle database. The product is based on a graphics subroutine package. SQL*Graph also supports color.

DISTRIBUTED DATABASES. SQL*Star, announced in 1986, is Oracle's architecture for distributed processing and distributed databases. It consists of two products: SQL*Net and SQL*Connect.

The SQL*Star architecture provides two major advantages. One is location transparency. The user does not have to know where a table is stored (locally or remotely) in order to access it. SQL*Star also allows the user to access multiple databases located on different machines (nodes in the network) in a single query. In other words, SQL*Star supports joins and unions across network nodes. You cannot yet update multiple databases in a single transaction, but, like all the vendors, Oracle plans to incorporate this in the future. In addition, Oracle plans to implement partitioned database tables, where rows of a single table can be distributed across different nodes, and coordinated updates of all copies of a table across nodes.

A distributed Oracle network requires SQL*Net, the Oracle RDBMS Version 5.1, SQL*Plus, and the appropriate communications protocol.

SQL*Net. SQL*Net is the basis for distributed processing in the Oracle environment. It allows an Oracle application running on one computer to access multiple remote Oracle databases. SQL*Net has been ported to the same environments as Version 5.1 of the Oracle RDBMS, and protocols are available to support several different networks. These include DECnet, Transmission Control Protocol/Internet Protocol (TCP/IP), asynch, and 3270.

In the Unix environment, SQL*Net supports connections between a Unix system and other systems via TCP/IP and asynch. Oracle plans support for other types of network connections in the future, including synchronous, SNA/LU6.2, and Manufacturing Automation Protocol (MAP) where appropriate.

SQL*Connect. SQL*Connect, which is not available yet, is Oracle's gateway to non-Oracle databases. Built on top of SQL*Net, SQL*Connect allows the user to log onto a non-Oracle database running on a remote host and to issue SQL commands against the foreign database. Initially, SQL*Connect will support access to IBM's two SQL-based DBMSs, Database 2 (DB2) and SQL/DS. Subsequent releases of SQL*Connect will feature access to non-SQL-based data files such as IMS/VS and VSAM.

In fact, a user with Networkstation Oracle on a PC and SQL*Connect could access DB2 directly. The Oracle RDBMS is not required. Oracle obviously would prefer that its customers acquire the entire Oracle package.

PC PRODUCTS. Version 4.0 of Oracle, requiring 640K of memory, has been available on the IBM PC since 1984. Oracle

has since introduced two subsequent versions of its mini and mainframe products: 5.0 added significant functionality, and 5.1 added distributed capabilities. Earlier this year, Oracle announced a set of three products for the IBM PC and PC LAN environments to bring PC-based Oracle up to date.

Professional Oracle. Professional Oracle implements Version 5.1 of Oracle on the PC. Since 5.1 would not gracefully fit within the DOS 640K memory box, Oracle chose to take advantage of the protected mode/extended memory features of the 286/386 PCs. Professional Oracle is the first DBMS to do this. And Oracle is marketing the product intelligently. Since the average PC out there still doesn't have memory beyond 640K, the company is combating both hardware and price barriers by offering the software packaged with extended memory boards at an attractive price. One package is designed specifically for an XT, upgrading the XT with a 286 processor. As more and more high-end PCs are acquired with 1 to 2MB of memory, this problem will disappear.

While IBM did the original research defining the relational model and SQL, Oracle was the first commercial product using the IBM blueprint.

Professional Oracle runs under DOS 3.x on an IBM-compatible 286, 386, or PS/2 (Model 50 and up). It requires 1.5MB of memory and a minimum of 6MB of disk storage. The product uses the memory above 640K for the DBMS code and data, storing only an 80K Oracle Executive within the 640K managed by DOS. The Oracle Executive is essentially an extension to DOS 3.x, managing the necessary transitions between protected mode (DBMS operation) and real mode (DOS operation). Oracle plans to adapt the product to OS/2 when it is available.

With Professional Oracle, developers can use the PC for applications development and port the application to the mini or mainframe.

Networkstation Oracle. Networkstation Oracle allows the PC to act as the front-end processor on a distributed Oracle network. It runs on any IBM-compatible PC with 512K memory.

PC LANs. Scheduled for release by the end of this year, LANserver Oracle allows a 286, 386, or PS/2 PC on a PC LAN to act as a dedicated Oracle database server (back end). It works in conjunction with Networkstation Oracle (or Professional Oracle with a networking option), which must be installed on each user workstation on the LAN to access the database server. The first release of LANserver Oracle will support TCP/IP on a Novell Advanced NetWare Ethernet LAN.

Product Positioning

Oracle's product and marketing philosophy is based on three major tenets: compatibility, portability, and connectibility. These have been the building blocks for product development, and Oracle feels that it has made considerable improvements in one or more of these areas every year. (Generally, the company introduces a major release every year.) By adhering to a standard data language (SQL) and providing a portable implementation of that language across a broad range of platforms, the company intends to create a "standard software environment as a basis for meeting the computational and data needs of the 1980s."

COMPATIBILITY. SQL compatibility is one of the cornerstones of Oracle's strategy. The company is committed to both formal (ANSI) and de facto (IBM's SQL) standards in this area. While IBM did the original research defining the relational model and SQL, Oracle was the first to produce a commercial product using the IBM blueprint as a foundation. IBM has since followed Oracle's lead with two RDBMSs of its own: SQL/DS for its smaller mainframes running DOS/Virtual Storage Extended (VSE) and VM/CMS, and DB2 for the larger mainframes running Multiple Virtual Storage (MVS). SQL/DS appeared in 1982, and DB2, in 1984. (Oracle points out that two of IBM's other SQL products are Oracle developed. These are the RDBMS sold for the IBM System 88 or Stratus, and SQL/RT for the IBM PC/RT.)

Oracle's implementation of SQL is compatible with and extends IBM's implementation in SQL/DS and DB2. In addition to faithfully supporting SQL as originally defined by IBM, Oracle is also committed to maintaining full compatibility with IBM's DB2 at the application level. Oracle states that any application program developed for DB2 will also run unchanged under Oracle (but not necessarily the reverse). This goes beyond merely executing a given SQL statement in the same way. Functionality such as data types supported, return codes, and error-processing must be implemented consistently as well. Therefore, the company states, it treats any change to DB2 as a "severity 1 bug" in the Oracle product, a bug that must be fixed immediately to maintain compatibility.

It is important to note that the application developed for DB2 must be recompiled to run under Oracle, and that any DB2 database accessed by the application must be converted to an Oracle database. The DB2 application cannot run under Oracle while still accessing a DB2 database. (This capability will come with SQL*Connect.)

Oracle is also committed to the relational model as proposed by Dr. E.F. Codd and embodied in the evolving American National Standards Institute (ANSI) standards.

PORTABILITY. Oracle runs on a wide variety of hardware and operating system platforms. These include many Unix variations as well as proprietary systems. The company recognized early on that its major customers, primarily Fortune 500 companies, had heterogeneous systems resulting in incompatibility at the hardware, operating system, and application levels. People

spent an incredible amount of time and money moving applications between systems. Therefore, "the ability to have a DBMS ported to many machines with the same functionality and a compatible applications development environment" would be very valuable.

CONNECTIVITY. Connectivity has been a recent development with the introduction of SQL*Net last year. Oracle can connect different operating environments from PCs to minis to mainframes and provide portable applications across all of them.

One of the advantages that Oracle has over its competitors is that it has been firmly rooted in SQL compatibility since its inception and already runs in a broad range of environments.

Marketing Strategy

Oracle's primary market is the Fortune 500. It has built its products with these customers in mind. These are the companies most concerned with adherence to standards, and Oracle has been immensely successful in aggressively selling its SQL-based RDBMS. Oracle is also clinging to Big Blue's coattails with DB2 compatibility. According to Cleveland, "Lots of people are going to buy DB2 because it's IBM. IBM is really pushing DB2 as a replacement for IMS [IBM's popular non-relational DBMS], but DB2 does not have the application development and end-user tools that Oracle does." As we have mentioned, DB2 applications can be moved to Oracle. In the future, SQL*Connect will let a user apply the Oracle tools directly against IBM databases without having to convert the data first.

Last year, Oracle made what it calls a difficult decision. Driven by marketing concerns, the company introduced distributed processing with SQL*Star and SQL*Net. The company actually would have preferred introducing performance features first to better compete in the transaction-processing environment. This is where it sees the predominant user requests—to better handle applications such as banking transactions and airline reservations systems, where a large number of users are generating a large number of transactions.

While continuing to enhance its distributed-processing products, Oracle is now focusing on two marketing thrusts: OLTP and end-user computing (the interface, integration, and consistency among modules). It has also discovered that adapting the European method of selling consulting and applications development is working very successfully in the United States as well. Oracle racked up $30 million in consulting revenues last year, up from none the previous year. The company sees the consulting approach as particularly important to its major customers—Fortune 500 companies involved in large IBM and DEC systems. And Oracle felt that it had to grow to a certain size before it could diversify into areas such as large-scale consulting. Oracle now has a consulting group of approximately 300 "and growing," says Cleveland.

THE COMPETITION. Both Oracle and Informix Software went public in 1986, while Relational Technology (Ingres) and Unify are still privately held. One of the advantages that Oracle has over its competitors is that it has been firmly rooted in SQL compatibility since its inception and already runs in a broad range of environments. Others are busy playing catch-up. Informix has invested in SQL compatibility with Informix-SQL, and is now concentrating on portability beyond Unix (read IBM's MVS and VM) and on improved performance (its recent Turbo version). Relational Technology entered the SQL fray late, announcing "broad" (but not complete) SQL compatibility with Release 5.0 of Ingres a year ago. The company is busy re-architecturing its product to use SQL as the access language to the database rather than QUEL, the original query language developed with Ingres.

Oracle's revenues are double those of its nearest competitor among these companies (Relational Technology). So Oracle can spend its significantly larger R&D budget (Oracle spends 20 percent of its revenues on R&D) on new performance enhancements, improved tools for both application development and end-user computing, support for bit-mapped terminals, et al. If all this is done properly, Oracle has the opportunity to maintain and extend its position as a market leader. As Cleveland puts it, the company is "well set up to address the needs of the market, and the bottom-line results reflect that." Having made the initial investment in SQL and the relational model, Oracle's point of view is that its competitors are now "ratifying what we have done all along."

THE UNIX MARKET. Oracle has made a significant turnaround in its perception of the importance of the Unix market. According to Cleveland, the level of commitment to Unix two years ago was "none, zero." Larry Ellison himself stated that the company would not focus on Unix until it saw the potential for significant returns. However, Unix soon became a strategic operating environment, benefiting from dramatic improvements in hardware technology.

Revenues from Unix sales are now about $18 million, up from $1 to $2 million two years ago, and the company anticipates $30+ million in Unix sales this year. As we have pointed out, all of Oracle's new product development is done in Unix now, rather than VAX/VMS. Unix is second only to VAX/VMS ($54 million) in revenues, and is the company's fastest growing product line. Oracle also states that its Unix products have the largest development and support group within the company.

Unix revenues break down as follows:

Direct sales (end users)	20-25%
OEMs	25%

VARs	5-10%
International	40%

DISTRIBUTION. The company sells the Oracle product line through direct sales offices, dealers, international distributors, original-equipment manufacturers (OEMs), and value-added resellers (VARs). Unlike some of its competitors, Oracle prefers that its OEMs port the Oracle products to their own systems. Oracle's 400 direct salespeople are located in 50 offices in the United States, Canada, Western Europe, Japan, China, and Australia.

SERVICE AND SUPPORT. The company recently centralized its support organization at its Belmont, California, headquarters. Support staff had previously been remotely deployed and associated with an individual sales office, an inefficient use of resources. Availability of support has been expanded, and under "premium support" services, customers can call in anytime and receive information on known bugs.

USING ORACLE: THE END-USER'S PERSPECTIVE

We evaluated Oracle Version 5.0 on an NCR Tower 32 under Unix System V. Since the current Oracle end-user tools such as Easy*SQL, SQL*Calc, and SQL*Menu are not yet available in Unix (they are in beta test and in the process of being ported), we will first describe the current state of the art in Unix from the end-user's perspective. Then we will give you a look at Easy*SQL, designed to make Oracle friendlier and easier to use. (We reviewed Easy*SQL in the DEC VAX/VMS environment.) These newer modules will be available in Unix by the end of the year.

SYSTEM REQUIREMENTS. Oracle requires, on average, 4 MB of memory for six users (an additional 80K of memory is needed for each user above six), 5 to 10MB of disk storage space, and support for shared memory.

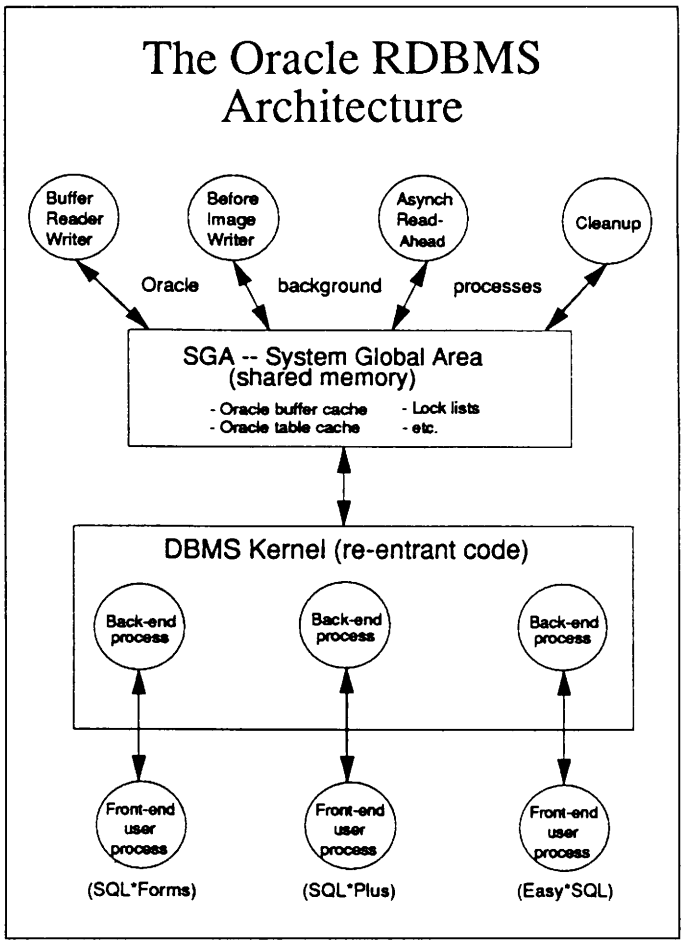
OVERALL ARCHITECTURE. Oracle was designed to take advantage of the following features to improve performance and data integrity.

Shared Memory and Reentrant Code. Oracle makes use of shared memory, setting up what it calls a System Global Area (SGA) in memory. When a user issues a request for data, Oracle first checks in shared memory (the SGA) to see if some or all of the data is already there. If it is, Oracle can minimize or eliminate the need to access the disk and, thus, improve performance. In addition, Oracle does not store a duplicate copy of the database kernel in memory for each user. All users can share access to a single image of executable code.

Write-Through Cache. Under the normal Unix filing system, when changes to a database are committed, they are written to the Unix buffer cache and then written to disk when Unix flushes the buffer cache. This can be done implicitly (when a certain amount of data has accumulated in the buffer or a certain amount of time has elapsed, say 30 seconds) or explicitly. With this method of writing to disk, if the system were to crash, any committed changes in the buffer cache would be lost. Yet the DBMS thinks that these changes have actually been made to the database. This can obviously affect the integrity of the database.

Oracle uses a special mechanism called "write-through cache" to flag changed (committed) data in the Unix buffer cache and to force the data to be written to the disk immediately. While this actually slows performance down a little, the company considers it an absolute requirement to guarantee data integrity.

It is important to note that write-through cache is only valid for those Unix operating systems that support it, such as those derived from AT&T System V. Some of the newer Unix operating systems, such as Dynix on Sequent, are also beginning to support write-through cache. Other systems must use raw-disk devices (bypassing the Unix filing system altogether) in order to guarantee data integrity.



The Oracle RDBMS uses a two-task architecture (splitting the DBMS into a front-end user process and a back-end database kernel process) and takes advantage of both shared memory and shared (reentrant) code.

Two-Task Architecture. Oracle separates the processing of its DBMS into a front-end user process and a back-end database kernel process. These foreground and background processes communicate using the Unix interprocess communication facility (IPCF); the mechanisms for this are pipes, messages, and shared memory. Normally, when data (e.g., rows in a table) are sent from the front end to the back end, each row is sent separately, requiring at least two system calls ("Here it comes" and "Thanks, I got it"). Therefore, for 10 rows, 20 system calls are executed. Oracle speeds up the process with an "array interface." Instead of sending rows one at a time, the system sends a package of rows all at once (the amount in a "package" is predetermined by the database administrator). This reduces the number of system calls and makes data transfer faster and more efficient. The array interface is part of the Oracle DBMS, not the Unix operating system.

No "umbrella" menu or shell exists that encompasses all the Oracle modules. Each module is invoked separately from the Unix prompt.

Oracle's two-task architecture also makes use of shared memory. Rather than moving data around in the system from process to process, the processes exchange semaphores (messages) identifying where the data is in memory.

Having a two-task architecture has helped Oracle in its transition to a networked environment, where the front end and back end must run on different machines to achieve performance objectives.

DATABASE PARAMETERS. Oracle allows 9,500 tables per database, unlimited rows in a table, up to 254 columns (fields) per table, and a maximum record size of 126KB. An unlimited number of indexes can be defined per table.

User Interface

No "umbrella" menu or shell exists that encompasses all the Oracle modules. Each module is invoked separately from the Unix prompt. Your username and password are required each time you call an Oracle module. (Your Oracle log-in can be made part of your Unix log-in to avoid this problem.)

Easy*SQL is an effort to provide a menu-driven user interface. SQL*Menu can also be used to create a custom menu environment for the end user. Until these products are widely available in Unix, the user is stuck with separate programs. Recognizing this inconvenience and lack of integration, Oracle is currently developing the means to enable the user to call any Oracle module from within any other module.

SQL*PLUS. The primary interface to the Oracle RDBMS on the Unix system is SQL*Plus. This interactive, command-driven interface is the only way to create and modify tables, create indexes, grant access permissions to the database, etc. SQL*Plus consists of SQL commands (e.g., create table, create index, select, insert, update, alter table) and extensions to SQL called SQL*Plus commands (hence, the name for the product!). SQL commands work against the database, and SQL*Plus commands are for formatting results, setting options, and editing and saving command files.

You enter commands at the SQL> prompt. SQL commands can span multiple lines (as you will see in several of the illustrations in this article) and must always end with a semicolon. Oracle is quite forgiving here, giving you another input line if you forget the semicolon. (It assumes you are not finished entering the query.) We forgot the semicolon often and appreciated not having to start over. SQL*Plus commands do not require a semicolon but must be entered on a single line.

The current SQL command is always stored in the SQL buffer. You can edit the contents of the SQL buffer using a rudimentary line editor or with the Unix system editor (without leaving SQL*Plus).

One frustrating aspect is the fact that SQL*Plus commands (i.e., commands that are an extension to SQL) are often treated differently than standard SQL commands. SQL*Plus commands are not stored in the standard SQL buffer. If you make a mistake, you cannot just edit the commands; you have to reenter the entire command. If you wish to save and/or edit SQL*Plus commands, you must first move to another buffer area, issue an "input" command, and then enter the SQL*Plus commands. They will now be stored in this buffer and will be accessible for editing. In addition, when storing a series of commands in a file that contains any SQL*Plus commands, you cannot retrieve the file into the buffer and run it (which you could do if the file consisted of only SQL commands). You must use the "start" command. We found this somewhat confusing, and the documentation does not explain why these two types of commands are manipulated differently.

Although SQL*Plus is command driven, we found it relatively easy to learn and use. One negative factor we should point out, however, is a frustrating measure of inflexibility. If you misspell a command (e.g., enter "selct" instead of "select") and try to execute it, Oracle rebuffs you with an error message: "unknown command 'selct'—rest of line ignored". It does not retain the command in the buffer for editing, and you must reenter it. This is not fun for long, multiline commands.

On our system, Oracle did not automatically pause at the end of the screen when displaying information, which we found frustrating. You can set up a pause by editing your login.sql file, but we think the default should be a pause after every 24 lines.

We also found a few little loose ends in SQL*Plus. Whether these exist only on the NCR port or across other systems, we don't know. The "clear screen" command does nothing as far as

we can tell. The documentation states that case is not significant when you enter the name of an object in an Oracle database. However, we found we had to enter some table/view names in uppercase in order to successfully query the data.

HELP. We are not terribly impressed with Oracle's Help function. In SQL*Plus, Oracle provides online Help displayed on a full screen. We would prefer the use of windows to maintain context for the user (this is coming in the future). Help is not context sensitive, and there is no Help function key. (In fact, there are no function keys at all in SQL*Plus.) You get help information by entering the "help" command (for a list of all commands in SQL*Plus) or "help" followed by a specific command or keyword (such as "create table" or "select").

While Oracle's file management system is difficult for an end user to understand, it provides a great deal of flexibility for the developer.

The major negative of the Help function is the inability for the user to scroll freely through the Help text. The only control you have is to stop the text from scrolling forward while you read it. You cannot go backwards, nor can you selectively move from one topic to another within Help. You can only access Help on one topic at a time. As a result, we did not refer to the online Help very often, finding the documentation sufficient to answer most of our questions.

In SQL*Forms, there is no Help function to guide the forms designer through the design process with the exception of a full-screen display of all the function keys available. The same is true for the person using a form. And we might point out that this display of function keys is in no particular order, which doesn't help you find what you want quickly. In Easy*SQL and SQL*Calc, Help is context sensitive and accessed by a function key.

Customized Help can also be developed.

ERROR-HANDLING. Error messages were not always as clear as we would have liked from the end-user perspective. If you try to access a table for which you are not authorized, the message states that the table is not valid. We would prefer that the message told the truth: that you aren't authorized access to the table. If you enter a duplicate value in a field where the value must be unique, you are told "Oracle error occurred while trying to update record." That could mean lots of different things.

Oracle does provide a separate, clearly organized manual explaining error messages.

DOCUMENTATION. We like the Oracle documentation. The manuals are well written, clear and straightforward, and easily understood by a new user. Generally, they make good use of illustrations, spacing, graphics, bullets, etc. Jargon is defined, and the documentation often explains why something is done a certain way. For example, the manual clearly explains the trade-offs between using SQL*Plus and Easy*SQL. We especially liked encountering little glimmers here and there of a sense of humor on the part of the writers in some of the examples. Very refreshing.

The documentation comes in the form of many separate booklets. Each module has at least one and possibly up to three or more of these—for example, SQL*Forms has a Designer's Tutorial, a Designer's Reference, and an Operator's Guide. In addition, each module has a quick reference card. The manuals are a handy size for using and carrying around (unlike those we got for Ingres).

What we don't like are the following features: The manuals are bound, and you cannot open them flat. This is a problem for any reference manual, but particularly for those you want to use while keyboarding. We would prefer manuals with standard PC-style binding: plastic rings or 3-hole punched to fit in a looseleaf binder. There is no overall index covering all topics in all manuals. There is no "read me first" guide, something the end user would know to look at first. This would lead the user to the SQL*Plus User's Guide which starts at the beginning of Oracle and guides the user through a tutorial of sorts. The visual appeal of the SQL*Plus User's Guide would benefit from a cleaner typestyle, proportional spacing, and graphics.

Database Design

All tables are part of one database in Oracle. There is no concept of creating separate databases for different applications (e.g., a customer database, an inventory database, an employee database). So you need not create a database in Oracle before you can create tables. When Oracle is installed, a standard database file is created called oracle.dbs. When you create tables, they are all part of this database.

FILE MANAGEMENT. An Oracle database is stored as one large Unix file on the disk. Tables are not stored as discrete files. Thus, Oracle is responsible for opening and closing tables and managing the information in the database.

You can create additional databases, and Oracle also offers partitions as a way to segment databases. You might want to consider using partitions for performance reasons. Partitions would allow you to store specific tables in the database on different disks. Oracle states very clearly that its product is not designed for the unsophisticated user. While its file management system is difficult for an end user to understand, it provides a great deal of flexibility for the developer.

Unify also stores a database as a single file, while Informix-SQL and Ingres store tables and other database objects as separate files. The trade-offs here include faster operation by

```
SQL> create table subscription
  1    (acctno char(6) not null,
  2    pubcode char(2),
  3    origdate date,
  4    begindate date,
  5    enddate date,
  6    numberyrs number,
  7    amount number(9,2));

Table created.
```

*Here is how a table is created in Oracle using an SQL command in SQL*Plus. The account number field, designated as "not null", cannot be left blank when entering data.*

bypassing the Unix (operating system) file manager versus the need to recover the entire database even if only one small part becomes corrupted.

DATA DICTIONARY. Oracle stores all information about tables, columns, views, indexes, users, access privileges, and storage allocations in the data dictionary. As in any true RDBMS, the data dictionary is a set of two-dimensional tables that are accessible with the same SQL commands used to query user-created tables.

CREATING TABLES. Tables are created with the "create table" SQL command in SQL*Plus. You merely enter the table name and each column name, followed by a type and length (if appropriate).

Oracle supports the following generic data types:

- Char includes up to 240 alphanumeric characters.
- Numeric includes up to 38 digits; you can specify the precision (total number of digits in the column) and the scale (the number of digits to the right of the decimal point).
- Date includes date and time values; the default format for a date is "dd-MON-yy" as in 01-AUG-87. Oracle provides several alternative date formats, but the date format can only be changed on a one-time basis in a query with a conversion function. We were quite disappointed to find that we could not change the default format for dates. We prefer to enter and view dates in mm/dd/yy format (09/11/87).
- Long is a special char data type allowing a column to be 64KB in length. There can only be one "long" data type per table, and there are limitations on the manipulation of data in this column (e.g., long columns cannot be indexed).
- Raw and long raw data types are similar to the char and long data types, but contain binary data, which is not interpreted by Oracle.

Within SQL*Forms, you can further define character and numeric data types. You can specify alpha (only spaces and lower/upper-case letters), integer (no decimals), money, and a number of other restrictions. However, these restrictions only apply to data entered on the form, not data entered with SQL statements in SQL*Plus.

NULL VALUES. Oracle supports the concept of a null value. A null value is different from a blank or a zero, indicating rather that a value does not exist for a particular column in a record. In fact, Dr. Codd prefers the term "missing information" to "null value", since a null is not really a value at all.

Null values are appropriate in certain situations. For example, let's say an employee table includes a field for commission, but only salespeople are eligible for commissions. Therefore, records for non-salespeople would contain a null value in the commission field. When listing the column, the null value would look as if it were a blank. The key is whether the DBMS recognizes the null value as such and handles it properly when performing calculations and returning query results. Asking for the average commission results in different answers, depending on whether rows with null values are counted or not. Oracle handles this properly by ignoring rows with null values in such calculations. Oracle also provides the following:

- The ability to define a column in a table as "not null"; this means a value must be entered in the column.
- A comparison operator for null values. You can request to see rows where the value in a column is null or not null.
- A function that translates null values into specified non-null values. For example, to calculate salary and commission for all employees, you would want to convert null values in the commission column to zero (otherwise, salary plus a null value returns a null value).
- The ability to have Oracle display null values with a specified value, such as "N/A".

VIEWS. Support for views is very important in RDBMSs. Views allow you to create different windows through which you see the data in the database. In Oracle, views are created with the "create view" command, which contains a "select" statement to define the view. For example,

```
create view dept10 as
  select ename, empno, job, hiredate
  from employees
  where deptno = 10;
```

This view displays a subset of both the fields and rows in the employees table. The view definition is stored in the data dictionary, and the embedded query is processed when the view is accessed. That is, the data is not stored two times, once for the

table and once for the view. Views are virtual tables, and are treated for the most part as if they were tables. They can be accessed and manipulated in the same way as the underlying tables (which are called base tables). Views can be queried, and, if the appropriate permissions are granted, the base table data can be changed through the view. Views can also be joined to tables and other views, and views can be defined in terms of other views.

Benefits of views include:

- Data independence. The underlying database structure can be changed, and the user doesn't need to be aware of this.
- Security. A view can be used to provide access only to specified columns and/or rows in a table.
- Simplicity—A view can represent a complex query, which is then accessed via a simpler query.

*One of the most powerful features in SQL*Forms is its ability to include "triggers" at the field, block, and form levels to perform additional validation and database processing.*

MODIFYING TABLE STRUCTURE. You can add or modify columns (change the data type or increase the size) in a table using the "alter table" command in SQL. However, there is no simple command in SQL for deleting or renaming columns or for reducing the size of a column. The only way to perform these is to essentially create a new table from the old one, using a nested query to select those columns desired from the old table.

To add a non-null column to a table after rows have already been inserted in the table, you add the column as accepting nulls, fill in values for the existing rows, and then modify the column to non-null status.

Creating and Modifying Screen Forms

Forms designed with the SQL*Forms module can be used to enter and edit data, perform queries, and build complex applications. SQL*Forms provides some of Oracle's most powerful and flexible capabilities.

USING SQL*FORMS. An SQL*Form consists of one or more "pages." Each page is a screen of data, and you can have an unlimited number of screens in a form. Each page then consists of one or more "blocks." Each block represents data from a single table and can display one or multiple rows of data.

The user interface consists of a combination of menus, fill-in boxes, and selection lists in overlapping windows. The first menu is the Choose Form menu, where you enter the name of the form you want to work on and select the desired operation (such as create, modify, run, etc.). You can access a list of existing forms, if necessary, and select one.

The first step is to create a form. After you enter the name of the form and choose "create," the Choose Block menu opens up, where you define each block on the form. You give each block a name, and can either design it from scratch (on a blank screen) or work from a default block. If you choose the default, a third menu opens up, where you identify the table (remember that each block represents data from one table), the number of rows from the table you want displayed at once, and the line on the screen on which the block should start. You then select the columns in the table to be displayed.

At this point, you can return to the Choose Block menu and define additional blocks if you wish to include data from other tables on the form. Next, you select "modify" on the menu to see the default form displayed. You are now in the screen painter and can customize the form by moving and sizing fields and labels, editing the text, and drawing lines and boxes. Once you get the hang of the function keys, it is relatively easy to create a customized format on the screen.

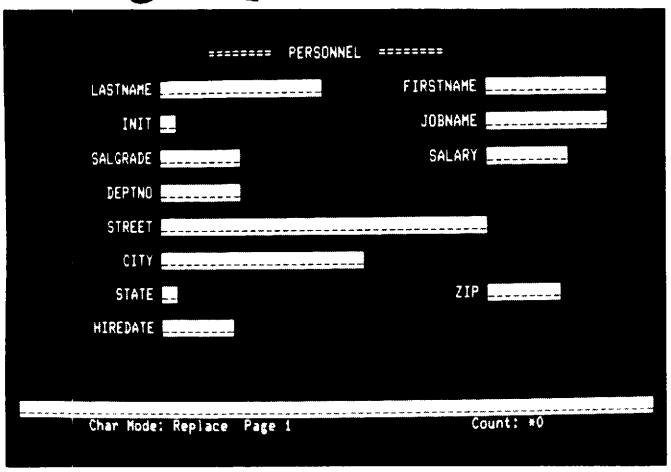
DEFINING FIELDS. You can also define attributes and validation criteria for each field simply by selecting items on a list in a window. Attributes include unique, display only, mandatory, uppercase, autoskip, and actions allowed in the field (input, update, query). You can also define a default value and a customized help message. Data validation can be very complex and sophisticated, providing referential integrity (a value does or does not exist in another table), a list of valid values (from another table), and a range of values. You can also connect master records to detail records on a multitable form.

DEFINING BLOCKS. At the block level, the forms designer can specify options such as the sequence number for blocks in a multitable form and default ordering ("where" and/or "order by" clauses that apply to all queries for the block).

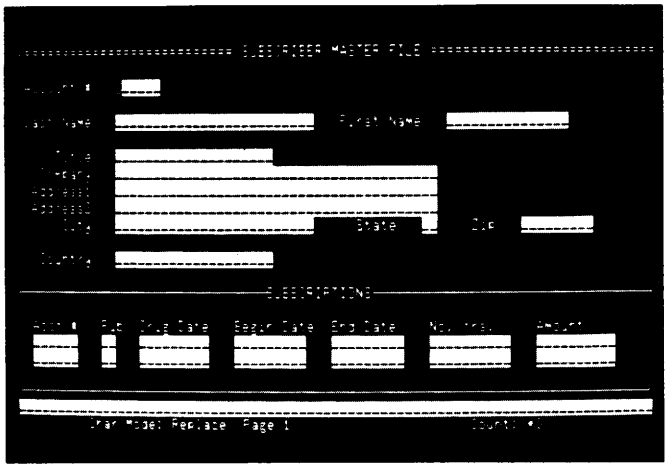
TRIGGERS. One of the most powerful features in SQL*Forms is its ability to include "triggers" at the field, block, and form levels to perform additional validation and database processing. Triggers allow the developer to build a form into an application. Triggers can be:

- SQL commands to query or modify any table that the operator has access to.
- Key triggers to attach macros to function keys (a macro, written in SQL*Forms commands, will be performed every time the operator presses a specific key)
- User exits to call a user program written in a 3GL

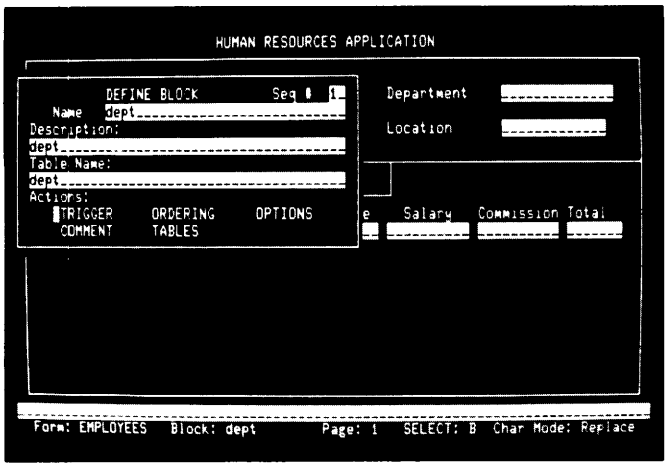
Using SQL*Forms



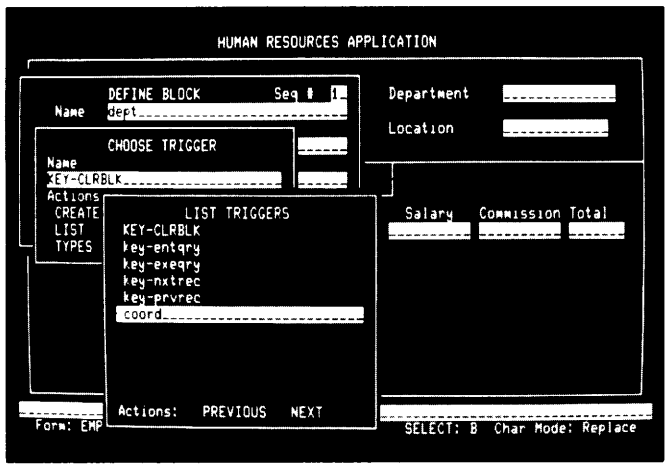
*This is a default form generated by SQL*Forms.*



This is a custom form we designed displaying data from two tables.



The user has the ability to define characteristics and processing for each block on a form. Here is the window showing the various options you have for defining a block.

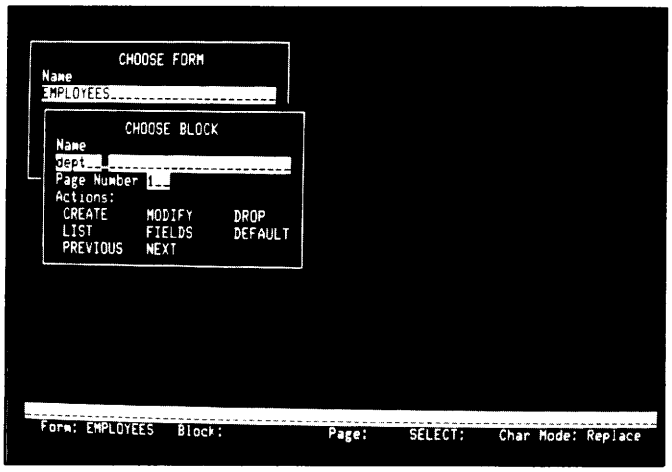


*One of the most powerful features of SQL*Forms is the ability to define "triggers" on a form. A trigger describes an action or series of actions that are to be taken depending on how information is entered on the form. Here is a list of triggers associated with the dept block on the Employees form.*

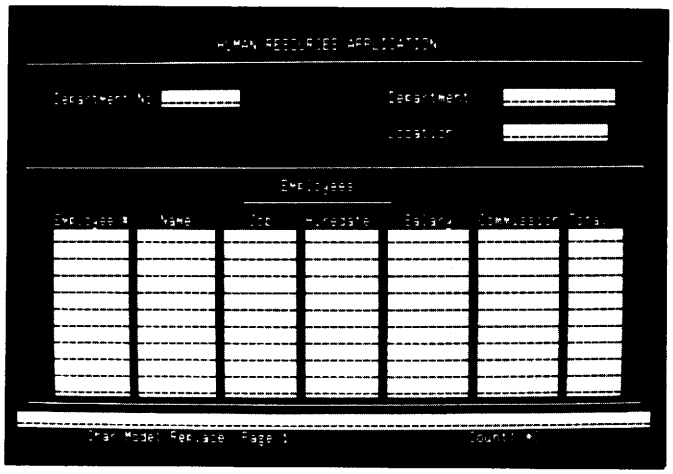
Triggers are associated with an event, such as the cursor entering or exiting a specific field or the operator pressing a function key. When you define a trigger, you must define its type—that is, the event that will cause the trigger to execute: pre-field, post-block, post-change, pre-query, etc. The choice of types varies depending on the level at which you are defining the trigger. When the event occurs, it sets off the trigger, and SQL*Forms will carry out the trigger steps. You also specify a message to be displayed if the trigger fails.

Triggers can be chained together (one trigger calling another) to develop very sophisticated and complex processing routines. For the most part, triggers, except for very simple ones, are designed to be used by the experienced applications developer, not the average end user. Triggers essentially involve programming and a clear understanding of the objective, implications, and appropriate timing of trigger steps.

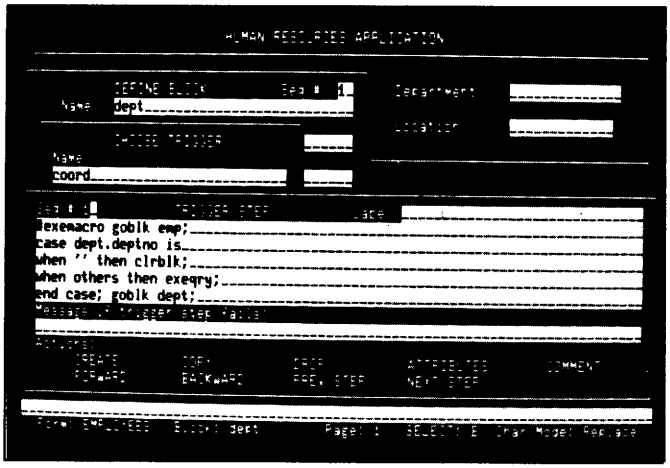
SOME SUGGESTIONS. Because SQL*Forms is so powerful and complex, it naturally takes some time and effort to master. In general, Oracle has done a good job on the user interface, the



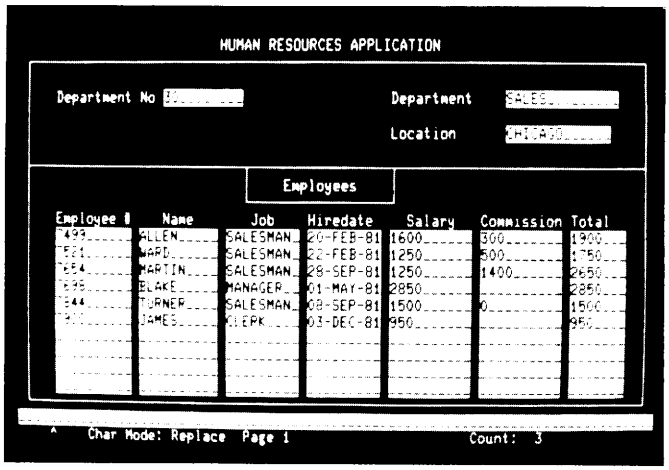
Once you've chosen a form, you then indicate which block on the form you want to work with (modify, etc.). A block designates a section on the form that contains data from a single table. A form can have multiple blocks.



*This Employees form is a custom form, designed with SQL*Forms, which displays information from two tables: the department (dept) table in the top block, and the employee (emp) table in the bottom block.*



Here is a sample trigger. The "coord" trigger allows the user to enter a value in the Department No. field. When the query is executed, the appropriate data will be displayed in both the dept block (displaying the department name and location) and the employee block (displaying the associated employees). This process is required to join two tables on a single form and to extract information from both tables in a single query.



This screen shows the results of the query for department number = 30.

incorporation of overlapping windows to maintain context, and the menus. But, as always, we have some suggestions for improvement. We don't like the order of items on the SQL*Forms menus. The items selected most frequently, such as modify and run, should appear at the beginning of a menu. And we desperately want to be able to select an item by its first letter or some unique character string. It takes much too long to move through a nine- or ten-item menu one item at a time (even with the option to go in either direction). We also find it counter-intuitive to select the item we want to work on before choosing what it is we want to do.

SQL*Forms is somewhat inflexible in its keystroke requirement, thus creating confusion for the new user. You must use Enter to select a choice on a menu, Return to enter a filled-in box, and PF3 ("accept") to select an item highlighted on a list. We would like one key to do all of the above. On SQL*Forms, the backspace key is mapped to take you to the previous field, while in SQL*Plus, backspace is the familiar destructive backspace. We found this frustrating.

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> select * from emp
  2  where sal > &salary and deptno = &dept;
Enter value for salary: 1000
Enter value for dept: 30
old   2: where sal > &salary and deptno = &dept
new   2: where sal > 1000 and deptno = 30

                                  Date
Empl Name   Job       Manager     Hired        Salary Commission Deptl Projl
---- ------ --------- ------- --------- ---------- ---------- ----- -----
7499 ALLEN  SALESMAN     7698 20-FEB-81   1,600.00     300.00    30   103
7521 WARD   SALESMAN     7698 22-FEB-81   1,250.00     500.00    30   103
7654 MARTIN SALESMAN     7698 28-SEP-81   1,250.00   1,400.00    30   103
7698 BLAKE  MANAGER      7839 01-MAY-81   2,850.00                30   101
7844 TURNER SALESMAN     7698 08-SEP-81   1,500.00        .00     30   102

SQL>
```

This is an example of how variables can be used in a query. This query can be stored in a file using the "save" command, retrieved when needed, and run.

Data Entry and Editing

You can enter and edit data in a table in two ways: use SQL commands in SQL*Plus, or use a form created in SQL*Forms. In SQL*Plus, data is added and updated with the standard insert, update, and delete SQL commands. This method for maintaining data has some limitations from the end-user perspective. First, you cannot add multiple records to a table in a single SQL command. However, you can use parameters (variables) in an SQL command to easily enter one record after another without having to reenter the entire SQL command. We found this feature very nice and easy to use.

Another limitation is the inability to create and apply extensive validity checks on the values in a column within SQL statements. Oracle does automatically ensure that data entered is the right data type and length, that all mandatory (not null) fields are entered, and that uniqueness is maintained where specified in an index. But no other validity checks, such as ranges for a value or a list of acceptable values, are available. This is one of the reasons for the following comment from an Oracle developer: "SQL*Plus can be very dangerous in the hands of end users without proper precautions (for example, restricting users to query-only in SQL*Plus)."

SQL*FORMS. Entering and editing data on a form created in SQL*Forms solves the problems described above. As we have mentioned, on a form, you can include extensive validity checks for entering and updating data. However, these checks do not affect data entered via SQL commands. So the Oracle RDBMS has no central repository of data validation and integrity rules. These rules must be included at the application level (in a form created with SQL*Forms or in a program written in a 3GL).

EXTERNAL FILES. The Oracle Data Loader (ODL) is a utility that can read external (non-Oracle) flat files in a sequential, fixed- or variable-length format. This utility is designed for a developer or very experienced user.

Indexing

Indexing in Oracle is straightforward and flexible. Indexes are standard B-tree indexes generated with the SQL command "create index". Once created, an index is automatically maintained. Use of indexes speeds performance in queries, especially those requiring joins, and can prevent duplicate values in a field when an index is specified as unique. The number of indexes for a table is unlimited, and indexes can be created or dropped at any time. The index key can be based on multiple columns but cannot be longer than 240 characters.

The existence of indexes is transparent to the user. Oracle automatically keeps track of what indexes exist and uses them as necessary in optimizing query processing. The user never needs to specify the use of an index when creating a query statement.

Queries

QUERY-BY-EXAMPLE. A form designed in SQL*Forms can be used to query the database. You "run" the form from within SQL*Forms to display it, hit the "query" function key to indicate that you wish to conduct a query (since the default mode on a form is entering new records—we think it should be the query function), and enter your criteria. You can enter an exact match or an SQL "where" statement in any of the fields. Unfortunately, SQL*Forms does not tell you immediately how many records it retrieved. You must scroll through all of them to get to the final "count" number on the screen or use a function key to get this.

SQL*PLUS. In SQL*Plus, you query the database using the standard SQL "select" command. SQL*Plus does tell you how many rows were retrieved in a query. As we have mentioned, you can edit the current query (the one stored in the SQL buffer), and save and retrieve repetitive queries. Stored queries can be executed at the time you enter SQL*Plus or at any time from within SQL*Plus with the "start" command. Stored queries can also include variables, or parameters, that can be entered by the user when the query is run. This is particularly helpful in situations such as when you need to insert many new records into the database, or if you wish to see the results of different sets of values in the "where" clause.

One feature we really like and found easy to use is the "column" command. This allows you to define both a format and a heading for a column in a list of query results. The default heading is the column name as it appears in the database, all in caps, as in DEPTNO. You can change this to something more readable, like Department # or Dept #, or even have the heading span multiple lines. You can also use templates to format numbers (e.g., $9,999.99) and dates, and to specify the width of a column.

A column command is independent of the table; the command applies to all instances of that column name, even if it appears in multiple tables. This command stays in effect until

you exit from SQL*Plus or change the column characteristics. To avoid having to reenter column commands over and over, you can store them in a file and have the file executed automatically at the beginning of each SQL*Plus session. One feature we would have liked but couldn't find anywhere is the ability to list the current settings for column headings and other parameters in SQL*Plus—a status screen of some sort.

The number of tables that can be joined in a single query is unlimited. And tables can be joined based on criteria other than exact matches. This is not true in many other RDBMSs, such as Ingres, Informix, and Unify.

OUTER JOINS. Oracle also supports "outer joins," allowing the query results to include unmatched rows from a table in a join. An example would be a request for a list of employees showing employee name from the employee table, and department number, name, and location from the department table. The two tables would be joined on the field department number, which appears in both tables. If there is a department in the department table for which there are no employees, a database that does not support outer joins would not return this department number in the query results. In Oracle, you can specify an outer join and see departments for which there are no employees. Outer joins are an extension of the standard SQL language.

SOUNDEX FUNCTION. The soundex function converts character values to their phonetic representation, allowing Oracle to accommodate spelling variations in query results. This is especially helpful in searching for names. The function "soundex(alin)" would match with Alan, Allan, Allen, and Allyn.

HIERARCHICAL PROCESSING. Oracle can interpret the data in tables that contain hierarchical relationships. An example here is generating an organization chart of sorts from an employee table. If each employee record contains the employee number of his or her manager and if you tell Oracle which employee is at the top of the chart, it can figure out who reports to whom and display this in a formatted list.

Generating Reports

SQL*PLUS. The SQL*Plus module offers some relatively easy-to-use formatting commands for simple reporting requirements. In addition to the "column" command we have already described, you can define "break on" and "compute sum" commands to calculate totals and subtotals on one or more columns. SQL*Plus reports provide centered page-headings and automatic page-numbering and date.

Unfortunately, when you run a report and include such formatting commands, they are in effect until you change or clear them. Again, a way to see what the settings are and to easily change or clear them would be immensely helpful. Oracle could do the end user a great favor by including some easy-to-use ring/pull-down menus in SQL*Plus for this purpose.

SQL*REPORT. The SQL*Report comes as part of the Oracle RDBMS. Called RPT/RPF, it is a report-writer similar to those found in the other RDBMS products we have previously reviewed. It provides flexibility, but you must write a program to generate a report. We believe that end users want a tool that is both easy and flexible, something that fills the large gap between SQL*Plus and SQL*Report. Easy*SQL offers some of these capabilities.

Customizing Applications

Oracle's tool for applications development is SQL*Forms. (SQL*Menu and SQL*Design Dictionary are also helpful, but do not run in the Unix environment.) While SQL*Forms is not really a fourth-generation language (4GL), it provides the essential capabilities of one. SQL*Forms has been designed to enable the developer to write applications without programming using triggers, which we have already described (see "Triggers" above). However, as we have seen, triggers are, essentially, a

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> get colheads
  1  column empno heading Emp#
  2  column ename heading Name
  3  column job heading Job
  4  column mgr heading Manager
  5  column hiredate heading DateHired
  6  column sal format 9,999.99 heading Salary
  7  column comm format 9,999.99 heading Commission
  8  column deptno heading Dept#
  9  column dname heading Department
 10  column loc heading Location
 11  column pname heading ProjectName format a12
 12  column projno heading Proj#
 13  column budget heading Budget format 99,999,999.99
 14* select * from emp
SQL>
```

*Here are some sample SQL*Plus commands for formatting query results.*

Emp#	Name	Job	Manager	Date Hired	Salary	Commission	Dept#	Proj#
7369	SMITH	CLERK	7902	17-DEC-80	800.00		20	101
7499	ALLEN	SALESMAN	7698	20-FEB-81	1,600.00	300.00	30	103
7521	WARD	SALESMAN	7698	22-FEB-81	1,250.00	500.00	30	103
7566	JONES	MANAGER	7839	02-APR-81	2,975.00		20	101
7654	MARTIN	SALESMAN	7698	28-SEP-81	1,250.00	1,400.00	30	103
7698	BLAKE	MANAGER	7839	01-MAY-81	2,850.00		30	101
7782	CLARK	MANAGER	7839	09-JUN-81	2,450.00		10	101
7788	SCOTT	ANALYST	7566	28-MAY-87	3,000.00		20	101
7839	KING	PRESIDENT		17-NOV-81	5,000.00		10	102
7844	TURNER	SALESMAN	7698	08-SEP-81	1,500.00	.00	30	102
7876	ADAMS	CLERK	7788	01-JUL-87	1,100.00		20	101
7900	JAMES	CLERK	7698	03-DEC-81	950.00		30	102
7902	FORD	ANALYST	7566	03-DEC-81	3,000.00		20	101
7934	MILLER	CLERK	7782	23-JAN-82	1,300.00		10	102
7999	BROWN	CLERK	7902	29-JAN-83	800.00		10	102

15 records selected.

SQL> SQL>

The results of a simple SQL query showing customized column headings.

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> get samplerpt
  1  column deptno heading 'Department'
  2  column ename heading 'Employee Name' format a20
  3  column sal heading 'Monthly Salary' format $99,999.99
  4  column comm like sal heading 'Monthly Commission'
  5  ttitle center 'Acme Widget Sales Dept. Personnel Report' skip skip
  6  break on deptno skip 1
  7  compute sum of sal on deptno
  8  select deptno, ename, sal, comm
  9  from emp
 10  where deptno in (10, 30)
 11* order by deptno
SQL>
```

*Here are the commands required for generating a simple report in SQL*Plus.*

```
                    Acme Widget Sales Dept. Personnel Report

Department Employee Name        Monthly Salary Monthly Commission
---------- -------------------- -------------- ------------------
        10 CLARK                      $2,450.00
           KING                       $5,000.00
           BROWN                        $800.00
           MILLER                     $1,300.00
********** ******************** --------------
sum                                   $9,550.00

        30 ALLEN                      $1,600.00            $300.00
           TURNER                     $1,500.00              $.00
           JAMES                        $950.00
           MARTIN                     $1,250.00          $1,400.00
           BLAKE                      $2,850.00
           WARD                       $1,250.00            $500.00
********** ******************** --------------
sum                                   $9,400.00

10 records selected.

SQL> SQL>
```

*The resulting SQL*Plus report.*

programming language. There are no "if-then" statements, for example, but equivalents exist in writing triggers. According to the company, this is where developers and users hit a "big wall." The company offers training classes to help developers over this hurdle.

Oracle maintains that SQL*Forms is at least as powerful, if not more so, than other vendors' 4GLs. The developers we interviewed attested to this. Both the company and the developers acknowledged that there is still a need to go outside of SQL*Forms (to a 3GL) to do some calculations. SQL*Forms is designed for forms-based applications, not applications such as CAD/CAM.

Database Security

As we mentioned before, a username and password ensure that an individual is an authorized Oracle user. Once in the DBMS, Oracle provides several levels of data security.

The database administrator (DBA), when creating a new Oracle user, gives the user one of three levels of access to Oracle:

- Connect access allows the user to log onto Oracle, query or update tables for which access has been granted, and create views; a "connect" user cannot create tables or indexes.

- Resource access includes connect privileges plus the ability to create tables and indexes and to grant and revoke privileges on these to other users.

- DBA access includes all of the connect and resource privileges plus the ability to access any data in the database, grant and revoke access privileges at the database level, create public synonyms, create and alter partitions, and perform full database exports, as well as other functions.

The user who creates a table or view in the database owns it. No one else can access or modify the data in the object unless the owner grants permission. Permission is granted in SQL*Plus with the "grant" command. An owner grants privileges (all, select, insert, update, delete, alter, index, and/or cluster) on a database object (to the column level) to a user (a specific individual or the public). Privileges can be extended "with grant option," which allows the user to, in turn, pass the granted privilege on to other users. The "revoke" command removes privileges from users, and, at the same time, from anyone to whom they have granted that privilege.

Access to specific rows (i.e., to specific data values) in a table is restricted through the use of views. For example, a view of a personnel table might specify that only rows for a particular department be included.

Database Integrity

TRANSACTION MANAGEMENT. Oracle recognizes transactions as logical units of work, providing both explicit and implicit "commit" and "roll-back" capabilities. Oracle stores a copy of data before it is changed in a Before Image (BI) file. This file is used if it is necessary to roll back an incomplete transaction.

If the system crashes, Oracle provides roll-forward recovery by keeping an After Image Journal (AIJ), a separate, parallel file of committed transactions (changes made to the database). The AIJ can be applied to a backup copy of the database to recreate the lost transactions and bring the database up to date. The trade-off of keeping an AIJ is the extra overhead required to write two copies of the changed data (Oracle estimates this as a 5 to 10 percent increase in overhead).

Oracle does not currently support incremental backup of tables—the ability to backup only changes rather than an entire table. This capability is currently under development and will be available in the next major release.

Easy*SQL

Now we will take a look at what the Easy*SQL product does for the end user to make Oracle friendlier and more PC-like, an

objective we think all of the mini-oriented DBMS vendors should espouse.

Easy*SQL, an alternative to SQL*Plus, is available only on VAX/VMS and the PC at the present time. Oracle expects to remedy this shortly by porting the product across all environments. While Easy*SQL provides much of the functionality of Oracle under a main menu, it does not include all Oracle capabilities. Missing are such things as indexing, subqueries, and the ability to grant and revoke privileges. We did not look at every function within Easy*SQL, but we will give you an idea of how the product works by describing the query, report, and table creation processes.

INTERFACE. Easy*SQL is menu driven. You use the up and down arrows to move among the choices on a menu, and the familiar message line at the bottom of the screen changes with each selection. You cannot choose a menu item by entering the first letter of the choice, a convenience we sorely missed in using the product (as we did in SQL*Forms). Any PC user will be frustrated by the time it takes to move a cursor through a long menu or list of items to get to the one desired. Easy*SQL limits the user to working with three tables at a time, but, for many applications, that is sufficient. You can also create views to work around this limitation.

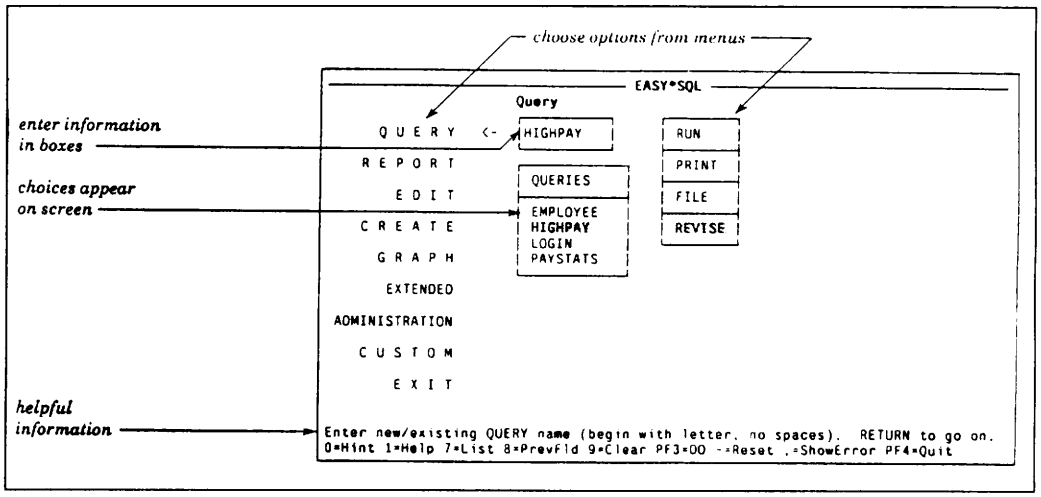
On the VAX, F1 is designated as Select, not Help, which doesn't bode well for PC users.

QUERIES. Choosing the Query option displays a box in which to enter a query name and a list of existing queries. For a new query, you then get a screen (the query panel) on which you enter the table(s) to be queried (up to three), and identify what column(s) to join on if there is more than one table. For each column to be selected, you enter the column name plus an optional format and heading. The next step is to enter your selection criteria (the "where" clause). You cannot use the "group by" or "order by" clause in query conditions, a negative in our estimation. To do this, you must generate a report in Easy*SQL. Nor can you include subqueries. (Subqueries require creating a view first or using SQL*Plus.)

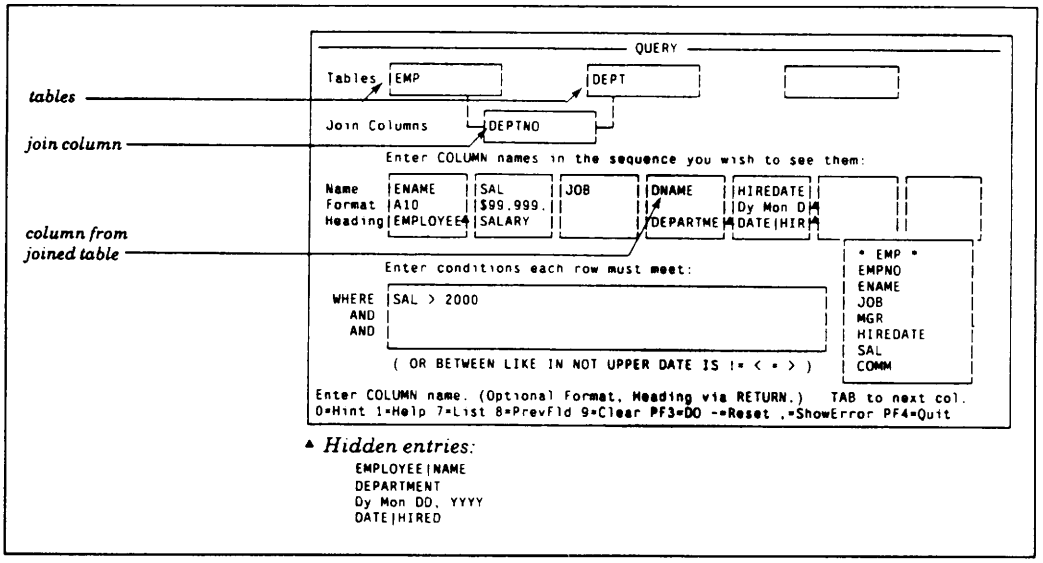
For existing queries, you may choose to run, print, file, or revise the query. Unfortunately, when revising a query, there is no quick way to get to an item in the middle of the screen without tabbing through the items in front of it. We found this very frustrating. Oracle needs to find a way to create a hierarchy of information on the screen to speed navigation to the desired item.

Some pluses:

- You can easily select all the columns for a query.
- You have great flexibility in formatting data.
- You can preview a query (Easy*SQL will display five rows for your review).



*Here is the Easy*SQL main menu showing the use of fill-in boxes and selection lists.*



*The Query panel in Easy*SQL. You indicate which tables to query (and the join column for more than one table), which columns to display, and which rows to display.*

Informix-SQL, Ingres, Unify, and Oracle: A Comparative Look

Feature	Informix-SQL	Ingres	Unify
Underlying file structure	C-ISAM	Unix file system	Unix file system or proprietary (option)
Database parameters			
Tables/database	No limit	No limit	256
Records/database	Limited by storage	No limit	2 billion
Fields/record	No limit	127	256
Record size	32KB	2KB	25.6KB *
User interface	Ring menus	Single-line menus	Full-screen menus
Menu bypass	No	Yes	Yes
Contextual help	Yes	Yes	Yes
Tutorial	No (demo database)	Yes	Yes
Data types			
Character	Yes	Yes (255 char. max.)	Yes (256 char. max.)
Numeric	Yes (small integer, integer, small float, float, decimal, serial)	Yes (integer—1, 2, 4 bytes; float—4, 8 bytes)	Yes (numeric, float)
Currency	Yes	Yes	Yes
Date	Yes	Yes	Yes
Time	No	Yes	Yes
Combination (links between 2 fields)	No	No	Yes
Binary	No	No	Yes
Variable length text	No	No	Yes
Field attributes			
Case conversion	Yes	Yes	No
Default value	Yes	Yes	Yes
Required field	Yes	Yes	No
Acceptable values	Yes	Yes	Yes
Verification (enter data twice)	Yes	No	No
Verify joins	Yes	Yes	Yes (explicit relationships)
Lookup joins	Yes	Yes	Yes
Composite joins	Yes	Yes	Yes
Formatting of data	Yes	Yes	Yes
Calculated fields	Yes	No	No
Display only (no entry/update)	Yes	Yes	No
Prompt (for data entry)	Yes	Yes	Yes
Error message	Yes	Yes	Yes
Customized help	No	Yes	Yes
Screen forms			
Default form generator	Yes	Yes	Yes
Customized	Yes	Yes	Yes
Multiple tables/form	Yes (14 max.)	Yes (10 max.)	Yes (single-screen limit)
Multiple screens/form	Yes	Yes	No
Embedded processing (if-then-else logic, display aggregates)	Yes	No (use 4GL)	No (but can be linked to SQL scripts)

* *Unify can create special parametric files to allow increased database parameters.*

Oracle
Unix file system or raw devices
9500
Limited by storage
254
126KB
Depends on module
No
No
Yes
Yes (240 char. max.; can have one field of 64K per table)
Yes (to 38 digits of precision)
No
Yes
No
No
Yes (Raw, and Long Raw)
Yes
Yes
Yes
Yes
Yes
No
Yes
Yes
Yes
Yes
Yes
Yes
No
Yes
Yes
Yes
Yes
Yes (no limit)
Yes
Yes

- You can define a query in Easy*SQL, store it (Oracle adds an extension of .sql), and run it with SQL*Plus.
- You can create calculated columns.
- You have access to many functions: sysdate (which is today's date), months between, power, round, square root, group functions such as average/count/sum/minimum/maximum, date arithmetic, and others.

Several more loose ends:

- Esc-K didn't work for us to show the function keys.
- The query panel doesn't display the name of the query you are working on.
- As in SQL*Plus, the user must press Stop and Resume to stop data from scrolling out of sight.

REPORTS. To sort, group, total, and/or format data into a report, you select the Report option. A report is based on a stored query, and you enter on the report panel only the additional information needed to create the report. All of the columns and rows included in the query will automatically be on the report.

You can enter a title and page footer and sort the report up to three columns or expressions (levels). The data will automatically be grouped on the first column and subgrouped on the second column. If you select columns to total, totals will correspond to groups, and subtotals, to subgroups. You can also get grand totals for all rows.

CREATING TABLES. For a new table, you define each column on the table by filling in boxes or selecting an option from a list. This includes the column name, data type, width, whether a value is required, a low and high value if appropriate (for range-checking), a default value, and a comment. The comment is used as a "hint" or customized Help when entering data in the field.

Oracle automatically creates an index on the first column in a table, so it is important to make this a primary key if possible.

OTHER FUNCTIONS. Within Easy*SQL, you can also create forms using a subset of SQL*Forms (or access forms designed with SQL*Forms), modify tables, create graphs, create a customized menu for accessing other modules of Oracle or other programs from within Easy*SQL, perform global updates and deletes, create views, link to Oracle on another computer, translate Oracle data into popular PC file formats (WKS, DIF, PRN, and SYLK), and backup the database.

SUMMARY. Easy*SQL has the potential to become the place where Oracle end users can happily hang out when not involved in a custom application. The menu provides access to a broad (although not complete) range of Oracle's functions with an easy-to-use and consistent interface. The *User's Guide* is very good; one of its strong points is explaining what Easy*SQL can't do and what the alternatives are.

ORACLE: THE DEVELOPER'S PERSPECTIVE

As usual, we interviewed experienced applications developers to get additional perspective on the Oracle RDBMS products. These developers have worked with very large databases and used Oracle in complex custom-programming environments.

Strengths

APPLICATIONS DEVELOPMENT. The developers are unanimous in citing applications development as one of Oracle's strongest points. Oracle is described as "a complete offering; what you need is there." This includes the capabilities of the core DBMS as well as the applications-development, decision-support, and end-user tools. Oracle has done a good job of building quality tools around SQL and the DBMS. Because of its commitment to the relational model and SQL from the beginning, the company has had the benefit of extensive experience from which to build solid tools.

SQL*Forms is "awesome," in the words of one of the developers; he has not yet run into anything SQL*Forms can't do. It is also a valuable tool for testing out SQL statements and prototyping a complex form or report. The ability to tune and do performance testing in SQL*Plus is a nice option. SQL*Plus is also good for linear (list-type) reports. "The DBMS is nice, being relational is nice, being SQL-based is nice, but Oracle's glory is in its development tools."

Informix-SQL, Ingres, Unify, and Oracle : A Comparative Look

Feature	Informix-SQL	Ingres	Unify
Query-By-Forms			
Exact match	Yes	Yes	Yes
Relational operators	Yes	Yes	Yes
Ranges	Yes	Yes	Yes
List of values	Yes	Yes	No
Wildcards	Yes	Yes	Yes
Maximum/minimum values	Yes (only indexed)	Yes	No
Print query results	Yes	Yes	No
Pass results to Report Writer	No	Yes	Yes
SQL			
Standard SQL statements:			
Data definition language (DDL)	Yes	Yes	No
Data manipulation language (DML)	Yes	Yes	Yes
Query language	Yes	Yes	Yes
Extensions to SQL:			
Commit/roll-back transactions	Yes	Yes	No
Execute operating system commands	No	Yes	Yes
Load/unload data to/from ASCII file	Yes	Yes	Yes
Additional data definition statements (alter, drop table, etc.)	Yes	No	No
Can be embedded in C/Cobol programs	Yes	Yes	Yes
Can create a new table with query results	Yes (temp. table)	Yes (permanent or temporary)	No
Stored queries	Yes	Yes	Yes
Case-insensitive (e.g., field names)	Yes	No	No
Optimizer	Yes	Yes	Yes
How create SQL queries	Interactive editor or Unix system editor	Interactive SQL/QUEL interface	Unix system editor
Report Writer			
Nonprocedural	Yes	Yes	Yes
Default report generator	Yes	Yes	No
Interactive report generator using screen forms	No	Yes (RBF)	No
Interactive debugging	Yes	No	Yes
Input source	SQL	SQL/QUEL	SQL, ASCII file, screen form, program
Multiple tables	Yes	Yes	Yes
Page formatting	Yes	Yes	Yes
Headers and footers	Yes	Yes	Yes
Data formatting	Yes	Yes	Yes
Sort data	Yes (8 levels)	Yes (unlimited)	Yes (unlimited)
Aggregate functions	Yes	Yes	Yes
Logical processing (if-then-else logic)	Yes	Yes	Yes
User variables	Yes	No	Yes
Prompt for input variables at run-time	Yes	Yes	Yes

Oracle
Yes
Yes
Yes
Yes
Yes
Yes
Yes
No
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Interactive SQL interface
Yes
Yes
No
No
SQL
Yes
Yes
Yes
Yes
Yes
Yes (unlimited)
Yes
Yes
Yes
Yes

PORTABILITY. All of the developers cited the portability and consistency of Oracle across many platforms. The breadth of the platforms supported by Oracle is real, not just part of the marketing hype. One developer designs systems on PC/ATs and ports them to the IBM mainframe, to DEC VAX/VMS, and to Unix. "The only thing that we have had to change is the representation of the 'not equals' symbol between the PC and IBM/VM. In .5MB of programs, we changed maybe 20 signs."

Another developer stated that Oracle's philosophy of DB2-compatibility has worked remarkably well, and that Oracle is the closest of the four rivals (Oracle, Ingres, Informix, Unify) to being a DB2 clone. This includes the way punctuation is defined and the way the system catalogs are set up. Oracle has been more careful than the others in this area.

NETWORKING. Oracle can, in fact, deliver a network connecting Oracle on the PC, the mini, and the mainframe. "It's there and it works." According to one developer, both Ingres and Oracle are at about the same stage of evolution in developing a network, although the market perceives Oracle as being ahead.

PERFORMANCE. Use of shared memory helps improve Oracle's performance. According to one developer, in some cases, 80 percent of database reads will find what they need in memory instead of having to access the disk. "The downside is that, on machines without lots of memory, Oracle tends to be slow." Developers are waiting for the new version of Oracle designed for "heavy-duty transaction-processing."

LEADERSHIP. Oracle is described as a leader in introducing new features. "Every new release has had major advances in performance, functionality, and tools. For developers, this is really key. We're in it for the long haul. Oracle has done a good job of keeping up and being a trend-setter, and it is still leading the industry."

Oracle's reputation as a savvy marketing organization was also mentioned as fostering the perception of the company as a leader. One developer described Oracle as a very aggressive marketing company that has become a "computer market phenomenon." The company's biggest pluses are its name, image, and the customer awareness it has in the marketplace. "Oracle keeps popping up everywhere; the company has done a phenomenal sales job. Oracle is the clear leader in the upper end of the market (DEC VAX, larger Unix systems). And people are buying the company more than they are buying the product." Oracle has also done a good job of marketing in Europe, having started this effort early.

SUPPORT. Support is described as "pretty good—comparable to everyone else" and "pretty good, considering the fact that Oracle runs in so many environments and supports so many operating systems." The recent centralization of the support staff is viewed as a positive step.

Weaknesses

REPORT-WRITER. The big weakness for Oracle is its report-writer (RPT/RPF). One developer described it as a 3GL report-writer—"It's not even close [in ease of use] when compared to SQL*Forms." All the developers agree that RPT is a problem for the average end user. It is also slow. Therefore, they do not use the report-writer, preferring to use SQL*Plus for simple reports and C programs for complex reports. Oracle is planning a new version of the report-writer, and the developers seem willing to wait for something better.

QUALITY CONTROL. One developer who has experience with several DBMSs states that Oracle has the "longest list of outstanding bugs and workarounds." One reason for this may be Oracle's decision to have many OEMs do their own porting of the product, which prevents Oracle from having the same degree of source code control it would have if it did its own ports. Though this OEM strategy allows Oracle to follow a more rapid growth path, the lack of control and the resulting divergence among versions of Oracle for different platforms offset that benefit.

PERFORMANCE. Like Ingres, Oracle does not perform as well on smaller boxes as it does on large, hefty systems. One reason for this is that both Oracle and Ingres were developed in big, virtual machine environments, and moving down to the Unix

Informix-SQL, Ingres, Unify, and Oracle : A Comparative Look

Feature	Informix-SQL	Ingres	Unify
B-tree indexing			
Max. no. indexes	No limit	No limit	255/database
Max. no. fields/index	8	10	8
Max. size of index key	120 chars.	2,000 chars.	240 chars.
Order options	Ascend/descend	Ascend/descend	Ascend/descend
Unique index	Yes	Yes	Yes
Database security			
Login password	No	No	Yes
Database-level access	Yes	Yes	Yes
Table-level access	Yes	Yes	Yes
Record-level access	Yes	Yes	Yes
Field-level access	Yes	Yes	Yes
Access by time of day	No	Yes	No
Access by location (workstation)	No	Yes	No
Ability to customize standard menus	No	No	Yes
Ability to design application menus	Yes (1/database)	Yes	Yes
Default menu generator	No	Yes	Yes
Custom help	No	Yes	Yes
Ability to create views of database	Yes (in data dictionary)	Yes	Yes (only in forms)
File access methods			
Hash indexes	No	Yes	Yes (primary keys)
Links (explicit table relationships)	No	No	Yes
B-trees	Yes	Yes	Yes
Buffered sequential	Yes	Yes	Yes
Transactions			
Logging	Yes	Yes	Yes
Commit/roll-back transaction	Yes	Yes	No
Roll forward	Yes	Yes	Yes
Referential integrity	Built in at application level	Built in at application level	Part of data dictionary
Concurrency control—locking levels:			
Database	Yes	Yes	Yes
Table	Yes	Yes	Yes
Record	Yes	Yes	Yes
Raw input/output	No	No	Yes
Database can span multiple physical devices (disks)	Yes	Yes	Yes
Networked version for IBM PC	Yes	No, but coming	Yes

Oracle
No limit
No limit
240 chars.
Ascend/decend
Yes
Yes
Yes
Yes
Yes
Yes
Yes
No
Yes
No
Yes
Yes
No
No
Yes
Yes
Yes
Yes
Yes
Built in at application level
Yes
Yes
Yes
Yes
Yes
Yes

supermicros has not been easy. In one developer's opinion, "while the 68020s are becoming a decent match for these two products, I would think long and hard before putting Oracle (or Ingres) on a machine like an Altos."

DOCUMENTATION. The good news is that the new version of the Oracle documentation is much better than past renditions. One developer cited the number of tutorials now available; another described the documentation as "excellent" from the end-user perspective. However, the documentation needs an overall index. Other developers expressed concern that, with respect to the technical aspects of Oracle, the documentation tends to make the product seem like a black box—a "put this in and get this out" approach rather than an explanation about what's going on. And the feeling is that the closer to the kernel you get, the less information there is.

DELIVERY DATES. Oracle has not proven to be consistently reliable in meeting delivery dates, particularly over the past year. Developers and OEMs tend to doubt the dates they are given from Oracle.

IMPORT/EXPORT. One developer mentioned the need for Oracle's utilities to provide the logic to import and export selected rows.

Summary

From the developers' point of view, Oracle has some impressive strengths: the completeness of its DBMS core, the quality of the tools surrounding the DBMS, the fact that Oracle runs on a broad range of platforms, its leanings toward user friendliness, and its sales and marketing success. The negatives are the report-writer (no surprise here; everyone else has the same problem), its performance on smaller machines, and some concerns about quality control and the ability to meet product delivery schedules.

From the end user's point of view, Oracle has a lot of pluses and a lot of potential. The functionality is certainly there, and the company is paying increasing attention to user interface issues. SQL*Forms is both powerful and relatively easy to use (up to a certain point of complexity). SQL*Plus, although it is command driven (and you still have to master the SQL language to use it), is relatively straightforward in its functionality. A menu/windowing overlay would be a welcome addition for the end user here.

In the Unix environment, Oracle appears to be in a transition phase. The newer end-user interface and decision-processing tools are not yet available, so there is no cohesive, overall environment for the end user. Once Easy*SQL, SQL*Calc, and SQL*Menu are available and there is a clean and easy level of integration among modules, Oracle will be a more comfortable and inviting environment for the end user. Easy*SQL in particular will solve some problems for the end user, such as making it easy to generate reports.

Oracle still has to improve both the interface and functionality for the end user. The following interface problems are not major, but eliminating them will measurably streamline the process for the user. Modifying tables is not as easy as it could be. Oracle does not keep the user as well informed as it should. And making SQL*Forms even easier to use for more complex applications will facilitate and encourage the use of Oracle for end-user applications development.

Futures

So far in this fiscal year, Oracle has been setting the stage for major expansion in all areas, especially the Unix marketplace. Oracle Version 5.1, replacing 5.0, has been ported to virtually all platforms; the process will be completed by the end of this year. This allows Oracle to introduce the SQL*Star distributed architecture across its operating universe. The company is also concentrating on porting end-user tools like Easy*SQL, SQL*Graph, and SQL*Menu to a broader range of environments. These are currently limited to DEC VAX/VMS (Easy*SQL is also available in the Oracle PC product). The objective is to establish a consistent platform with Oracle 5.1 and enhanced end-user tools. Since this process is almost complete, what's next for Oracle Corporation?

TRANSACTION PROCESSING. Last month, the company treated its user group to a preview of a new Oracle database kernel product, Oracle Transaction Processing Subsystem (TPS). We expect a formal announcement early next year and general availability by mid-1988. This major new release, which has been in development for two years (approximately 50

percent of the code has been rewritten), will significantly boost performance, allowing Oracle to compete more effectively in an on-line, transaction-processing (OLTP) environment. According to the company, Oracle TPS will not replace Oracle Version 5.1; the two database kernels will be developed in parallel. All of the current Oracle products will work with Oracle TPS as well.

With Oracle TPS, the company is packaging speed on top of its already established compatibility, portability, and connectivity. This enhanced performance comes from two developments. The first is the addition of different logging and compression mechanisms to achieve higher transaction rates. The kernel will now be able to write changes to the disk independent of the user application. The second is the ability to use multiple processors.

Oracle TPS sports a different architecture than the current product, one that the company claims will handle transactions faster and will minimize the traditional bottlenecks.

New Architecture. Oracle TPS sports a different architecture than the current product, one that the company claims will handle transactions 5 to 10 times faster than Version 5.1 and will minimize the traditional bottlenecks faced by a DBMS when it tries to process a large number of transactions simultaneously.

What are these bottlenecks? They are CPU performance, the disk subsystem (disk I/O), and the database kernel itself as it writes changes to the disk. According to Oracle, today's systems—with their multiple processors, improved processor performance, larger available memory, and new bus architectures with wide bandwidth to move larger amounts of data at once—have the capability to handle the first two obstacles to high performance.

Oracle, with its separate front-end and back-end processes, is well set up to take advantage of a multiprocessor environment, and the new architecture will facilitate this. The back-end processes can be spread out among multiple CPUs. Theoretically, if there are enough processors to go around, each user would get his own CPU to bash against. For example, in a Sequent system with 10 CPU boards, each of ten users could work on a separate CPU. With twenty users, only two would be on each CPU, and so on. All of the processors would access shared memory, cached tables, and programs. The new Oracle RDBMS kernel will also be multi-threaded, allowing it to handle multiple requests at once.

The third bottleneck, however, presents a different situation. Even with multiple users on multiple processors, each user of a database must still wait for the DBMS kernel to write changes to the disk before going on to the next transaction. Oracle TPS has been rewritten to add new concurrent logging mechanisms and writing mechanisms to allow the DBMS to process transactions continuously without having to wait to physically read and write the database. All logging and writing of committed changes is done independently of both foreground processes and the Unix operating system. The benefit is that the amount of time that a user process has to wait for a commit (a returned write process) has been significantly reduced.

When questioned about the impact of the new Sybase product, with its emphasis on OLTP, on the RDBMS market, the company diplomatically replied: "Sybase has raised everyone's consciousness that a good relational database can also have great speed. They have provided the 'proof statement' of this, which has been good for the market." However, Oracle states that it is pursuing capability that is significantly different from what Sybase has implemented, since Sybase doesn't currently address the use of multiprocessors (we expect Sybase to do so within the next year). Oracle considers the ability to use multiprocessors effectively and distribute the processing load of Oracle and its users onto different CPUs a key to excellent performance. Oracle also hopes that Sybase will not be able to compete over the long haul because of its late start and limited platforms. However, these platforms are three of the most powerful today—DEC VAX/VMS, Sun workstations, and Pyramid 9000s.

Programming Language. Oracle will also introduce a new programming language, PL/SQL, which it describes as a combination of PL/I and Ada. The company is reluctant to label PL/SQL as a 4GL; Oracle would rather call it a transaction-processing language. It will enable an application to package multiple transactions, send the package for processing, and receive back the appropriate rows of data. PL/SQL can be used in conjunction with SQL*Forms, the current Oracle application development tool.

DATA DICTIONARY. Oracle stated that it has not incorporated triggers and integrity rules into the data dictionary à la Sybase because the implications of doing so "have not been fully hashed out by ANSI. Sybase has implemented a proprietary solution to the problem." Oracle is stubbornly adhering to its commitment to standards.

USER INTERFACE. The company has adopted the XWindow standard, and all new products will include support for windows, bit-mapped workstations, and the use of a mouse. The first product to reflect these enhancements will be a new report-writer, due out in early 1988.

NEXT STEPS. Looking beyond transaction processing, the company described its next efforts (essentially "Version 6" of Oracle) as a combination of more performance enhancements and the implementation of the "other side of the distributed question," in particular, multi-site updates in a single transaction.

The Bottom Line

Oracle has obviously done a lot of things right, and the perception is that the company's RDBMS is out ahead of many of its competitors. Some of this is true, and some is just good marketing. We see the competition heating up in three major areas: on-line transaction processing, enhanced networking capabilities, and end-user application development tools. Oracle is well-positioned, with its current products and future plans, to continue its pattern of success and growth.

In spite of Oracle's predictions with respect to Sybase's future, we still think Sybase is worth watching over the next year. The company's approach to incorporating integrity rules and stored processes in the data dictionary has significant implications for developer productivity. And others agree with us. Pyramid Technology, with its powerhouse hardware systems, has just announced an OEM agreement with Sybase. Pyramid will be marketing its hardware with the Sybase RDBMS software as a complete transaction-processing package. The potential exists for Oracle to do the same thing with its new TPS database kernel. (Pyramid has an OEM commitment to Oracle as well.) And we cannot count Relational Technology or Informix out.

So the race is on. Oracle, Sybase, and Relational Technology appear to be the front-runners at the moment. Oracle's strategy will be to maintain a leading-edge approach to product development and make timely product introductions while effectively managing the rapid growth of the company.

•TUTORIAL•

System V.3 Streams

Unix V.3 uses streams to address the problem of changing the kernel design to allow better support to the device driver.

By Gary J. Nutt

In the August column, we described the Unix architecture as it relates to devices. By defining rigid interfaces among device drivers, the kernel, and applications programs, it is possible to add new devices to the kernel space.

This approach has not changed since the early architecture of Unix. However, it is now showing signs of age, particularly for character I/O devices (those which deal with characters rather than blocks of bytes). The designers of Research Unix decided to redesign the character I/O part of the architecture for Eighth Edition Unix. (For more information on Research Unix versus AT&T or BSD Unix, see Vol. 2, No. 7.) The new design and implementation have subsequently been introduced as part of AT&T Unix System V Release 3.

What was wrong with the old architecture? First, it essentially required that character I/O device drivers process a single character at a time. Secondly, the approach was not really robust enough to handle very sophisticated character I/O devices such as network interfaces. This led to awkward implementations which were not maintainable and which did not perform well. It is possible to improve driver performance by altering parts of the kernel in conjunction with the addition of specialized drivers. In the case of network drivers, the specialized drivers have become very large and complex due to the requirement to implement network protocols in the kernel space. This is not always a good approach for several reasons: The vendor that alters the code must own a source license; the vendor that alters the code creates a nonstandard version of Unix; and altering the code requires a substantial technology investment, since the changes are complex.

How is the V.3 kernel changed to accommodate more powerful drivers? The limits to the old design are the result of two fundamental problems: Driver code cannot make full use of the kernel commands since it is always operating "beside the kernel" rather than "on top of the kernel." And the general architecture is not conducive to constructing large, modular programs to implement a driver. The kernel design had to be changed to provide better support to the device driver and to allow the construction of elaborate drivers composed of many modules.

The System V.3 kernel addresses this problem by supporting streams. A stream is a full-duplex connection between an application program in user space and a device. Thus, a stream can be thought of as two chains of modules, one going from the program to the device (for writing), and the other going from the device to the program (for reading). Each module performs some specific type of processing on characters in its input queue, then puts the processed data onto the input queue for the "downstream" module in the chain. When a device is opened by an application program, a degenerate stream is built, composed of only a "stream head" module, which implements standard device functions, and a device-dependent driver. This configuration is equivalent to the old-style Unix driver, although the two modules interoperate with the kernel using a new interface. The stream can be changed dynamically by putting new modules between the stream head and the device. Thus, a device is opened as a dumb device, and then it becomes more intelligent as modules are pushed into the stream. Without going into detail, it is clear that the intent of the new architecture is to improve performance with manageable software. Thus, even though layers of software are introduced, the stream's architecture itself is designed to be very fast.

The beauty of streams is that they provide the means by which one can build a family of "device drivers" that can be adapted to different network controllers and still provide the same software interface to the application program. Furthermore, such implementations are executed in the kernel space, so they are faster than user-level implementations.

Let's consider a hypothetical situation to illustrate this last point. As a distributed application program developer (or end user), you require soft-

ware which runs on machines from several different vendors. You know that you need, as a prerequisite to distributed software, some form of interconnect among the machines so that they can transmit and receive data to and from one another. So you have installed a LAN which can be connected to each of your machines. Then you discover that none of your programs can communicate, even though your machines are interconnected, since the programs do not "speak" a common language. This is, of course, the network protocol problem. To solve this problem, you require a common, high-level protocol just below

AT&T System V.3 streams are a welcome change to Unix, encouraging open expansion of systems to accommodate multivendor configurations.

the application program level at each machine. You do not want to be tied to any particular LAN controller vendor since the technology—and pricing—in that market is rapidly changing. All of the controllers are intelligent, but some are more so than others. The stream modules can be used to provide a common interface to the application program. Let's assume that you choose to build your application programs on a remote procedure call (RPC) protocol. (This would be an approach similar to the one employed with the SNA LU6.2.) Then each system must support RPC in kernel space using controllers of differing capability. First, streams modules that implement various layers of the desired protocol are constructed. The modules that are closest to the device are system dependent; the modules closest to the stream head are independent of the system and may be reused at each different machine. Because large amounts of the protocol are independent, large amounts of the software are reused. In addition to the obvious economic gain (that of avoiding duplicate implementations), the resulting distributed system has much more consistent implementations at each node in the network.

AT&T System V.3 streams are a welcome change to Unix, encouraging open expansion of systems to accommodate multivendor configurations. Although they change the kernel to increase performance for ordinary character I/O, their real contribution is in their ability to accommodate a wide variety of network controllers working with a wide variety of network protocols. Although the original System V Release 3 did not provide many streams modules (only a few streams modules for 3BNet controllers), it did provide an architectural framework for extending V.3 to support networks. The big question now is whether the streams architecture is sufficiently general—and well enough built—to support real, multivendor distributed products. ◐

Gary J. Nutt is a professor of computer science at the University of Colorado in Boulder and president of Upside Technologies, a consulting firm also in Boulder.

The 1987 Seybold Executive Forum:
Visionary Management and Technology Directions in the Real World

The Program: The 1987 Seybold Executive Forum provides a context in which both business and information systems management can come together and build a common frame of reference about future directions in technology and strategic, corporate-wide implementations of information systems. The unqualified success of the Seybold Executive Forum format is due to three factors: Speaker are handpicked and their presentations carefully screened to avoid marketing hype. Presentations are limited to 20 minutes and are followed by an interactive dialogue between the speakers and Patricia Seybold. The audience joins a lively question-and-answer session following each presentation.

November 9th

Global & Organizational Issues

Patricia B. Seybold
President
Office Computing Group
"Leading Indicators in the Evolution of Information Technology"

Kenneth H. Olsen
President
Digital Equipment Corporation
"Building the Communications Infrastructure"

John Young
President
Hewlett-Packard
"Technology and People: America's Competitive Advantage"

Mitsuo Tada
Executive Vice President
Fujitsu America Inc.
"The Global Information Economy"

Chris Demos
Senior Business Advisor
Federal Express Corp.

Allen Krowe
Senior Vice President
IBM Corporation
"Adapting to Changing Technology"

Government Leader
"Aligning High Technology and Government"

Vittorio Cassoni
Senior Vice President
AT&T/Olivetti
"Partners in Progress"

Al Ramacciotti
President
American Airlines/Capture

Karen Vangorder
VP of Sales, Services and Distribution
Steelecase

Robert Reich
Author of <u>The Next American Frontier</u>, and <u>Tales of a New America</u>
"Meeting the Technology Challenge"

November 10th

Visionaries & Technology

Patricia B. Seybold
"Visionaries & Their Uses of Technology"

Bill Gates
Chairman & CEO
Microsoft Corporation
"Mission Critical Software"

Steve Keese
Consultant/ Systems Strategies
John Hancock Corporation
"Sowing the Seeds of Innovation"

Dr. Harry Tennant
Texas Instruments Fellow
Texas Instruments
"Strategic Implications of AI"

Liz Menten
Manager Information Systems
Pepsico Company
"Supporting End-User Computing"

Jean-Louis Gassée
VP of Product Development
Apple Computer
"Harnessing Renegade Computing"

Signe Weber
VP Planning & Technology
Shearson & Lehman Bros.
"Organizing for Change"

John Greenup
Senior Member of Technical Staff
General Electric

Carl Hodges
Director
Environmental Research Lab
"The Ultimate Beta Test: Biosphere II"

Dr. Joseph P. Allen
Executive Vice President
Space Industries, Inc.
"Expanding our Uses of Technology in Outer Space"

Richard Fox
Director of R & D
Walt Disney Imagineering
"Technology: Disney's Backstage Star"

The 1987 Seybold Executive Forum:
Visionary Management and Technology Directions in the Real World

November 11th

Technology
&
Teamwork

Patricia B. Seybold
"Creating & Maintaining High Performance Teams"

Fred Wang
President
Wang Laboratories
"Technology & Teamwork in the Office"

Marilyn Mantei
Senior Research Engineer
Center for Machine Intelligence
EDS

Jim Manzi
President & CEO
Lotus Corporation
"Enhancing Workgroup Productivity"

Martin Petraitis
Director of Automation
National Semiconductor
"People, Process, and Technology"

Ron Skeleton
Manager of Scheduling and Integration
Aerojet Strategic Propulsion

David Allen
Vice President
Insight Consulting
"Managing Accelerated Performance"

Joe Henson
President & CEO
Prime Computer
"Supporting High Performance in Manufacturing"

Registration Form

TO RESERVE YOUR SPACE AT THE *1987 SEYBOLD EXECUTIVE FORUM*,
SIMPLY FILL OUT THE ORDER FORM BELOW OR CALL US AT (617) 742-5200.

Discounts available for two or more attendees from the same company.
For information, call Debbie Hay (617) 742-5200.

Please register me.

- ☐ My check is enclosed for $1095. (Please make checks payable to Patricia Seybold's Office Computing Group)
- ☐ My purchase order number is _____
- ☐ Charge to my MasterCard/VISA (circle one)

Card# Expiration Date

Signature

Name

Title

Company

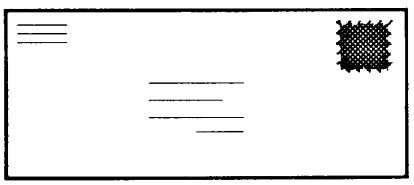
Address

City, State, Zip

Area Code, Phone Number

PLEASE RETURN THIS FORM FILLED OUT TO:

The Seybold Executive Forum
Patricia Seybold's Office Computing Group
148 State Street, Suite 612
Boston, Massachusetts 02109
(617) 742-5200



Cancellation Policy: Should a registrant be unable to attend the Forum, the Forum office will refund the full registration if notified before October 12th. Cancellations from October 12th to October 25th are subject to a $50 service charge. There will be no refunds as of October 26th. Substitutions may be made at any time.

NEWS ANALYSIS

PRODUCTS • TRENDS • ISSUES • ANALYSIS

• INSIDE •

• AFCAC •

AFCAC Challenged

The estimates of the value of the Air Force's AFCAC 251 bid have risen from $3.5 billion to as much as $4.5 billion, based on testimony in DEC's protest to the General Services Administration. The company had reason to be alarmed when the mammoth bid for office systems required a version of Unix that DEC neither had nor planned to support. With so much money at stake, DEC decided not to take it lying down. DEC's protest has officially suspended the AFCAC bid until administrative law Judge Vincent Labella rules on DEC's protest, probably in late October. DEC's source of irritation is that the bid specifically requires AT&T's System V version of Unix and requires that the operating system selected meet AT&T's System V Interface Definition (SVID) testing suite.

DEC contends that the bid's requirement of a specific vendor's iteration of Unix makes it a proprietary software bid and thus restricts competition. What would DEC like to see come out of the ruling? Simply that the Air Force spell out the specific functions that would be accomplished as part of the operating system. Sounds straightforward enough. However, if the bid is rewritten to DEC's liking, there would be no concrete way to prevent a vendor from bidding a proprietary operating system that could do everything that Unix could. Therefore, chances are good that DEC might be tempted to bid its mainstay VMS operating system. If Judge Labella rules in favor of DEC's protest (at least one vendor—Wang—has sided with DEC), AFCAC 251 may become a free-for-all.

The Air Force isn't taking DEC's protest lightly. It recently responded with the following statement:

The Air Force has a requirement for applications software portability today and in the future. The only currently available way to assure portability among different hardware is to ask vendors to supply an operating system that will easily adapt to the emerging IEEE standard, Posix. The System V Interface Definition (SVID) is the only generally available definition of the functionality required and is also the basis for the IEEE standard. It is also the basis for approximately 200 vendor operating system offerings including systems commercially available from Digital and Wang. Though some of those offerings require payment of a licensing fee to AT&T, Digital's own estimate is that it will amount to approximately one half of one percent of the estimated revenue of the contract. The National Bureau of Standards, the General Services Administration, and experts in the field agree that the Air Force's stated need is valid and will be satisfied by the specification of SVID. They have also agreed that the Air Force's intent to use the System V Verification Suite (SVVS) as a baseline to measure the degree of functional conformance to SVID is a rational, fair, and objective method of assuring future portability. ◐

—JSH

• IBM •

IBM Talks Unix

At a recent consultants' briefing, IBM revealed its long-term plans for its efforts with Unix. The most interesting of these included:

- To adopt Apollo's Network Computing System (NCS)
- To offer its AIX operating system across its systems lines from the PS/2 Model 80 up through its 9370
- To create bridges between Unix and its Systems Applications Architecture (SAA)

NCS AND DISTRIBUTED COMPUTING. IBM revealed its intention to adopt Apollo's Network Computing System (NCS). NCS allows for distributed processing within a heterogeneous network. In other words, through a Remote Procedure Call (RPC) function, a database or other program could be split among different vendors' CPUs. NCS also allows processes to be dynamically allocated to the CPU with available cycles. IBM will probably participate in the Network Computing Forum, an organization of 40 vendors promising to work to make NCS an industry standard.

The proposed move to accept NCS is not a radical departure for IBM, given the company's stated vision of its networking plans, especially where its RT/PC is concerned. IBM's own vision of distributed computing, Distributed Services, requires both local and remote transparency within a networked environment.

The initial departure from traditional time-sharing will take place within the Unix environment under IBM's version of Unix, AIX. However, IBM's vision of distributed computing goes beyond the Unix-based distributed RT network. In the view of Larry K. Loucks, member of IBM's senior technical staff and architect of the AIX operating system and the RT, the distributed network will become the equivalent of the supercomputer. "This will take all workstations and create a supercomputer out of them, with administrative management," he notes. He further believes that these distributed services will encompass remote and local file systems. The implementation of distributed services will take place at the operating system level, not the application level.

Transparency is the key to IBM's Distributed Services. In a recent paper, published in February 1987, IBM researchers noted that a local file and a remote file should be the same to the user. The concept of the "single system image" is the core of the networking. As a result, applications such as database and data management could run in this environment without modifications to existing object code. Another key component consists of remote file server characteristics such as remote file mounting (i.e., NFS). IBM also speculated that it will probably implement NFS sometime in the future. In the end, the client should not have to know that software resides on multiple servers. Therefore, a network can expand without the necessity of changing the system or software.

AIX DEVELOPMENTS. Initially, IBM took a low-key approach to Unix. However, as the market continues to heat up, IBM has begun pouring considerable resources into its version of Unix. While the original version of the operating system was purchased from Interactive Systems, IBM has taken charge of AIX, adding facilities and extensions. Most notable is IBM's plan to have a common version of AIX transcend its systems, from the PS/2 Model 80 up to the 9370. Xenix will be supported on lower-end models of the PS/2. IBM has based AIX on System V.1 with BSD 4.2 and 4.3 extensions. The long-term goal is to migrate to the Portable Operating System (Posix) standard when it is ready. IBM has added certain features of V.2, including a full screen editor and queueing. It does maintain a BSD version of Unix for the university community.

Windowing is an integral part of IBM's approach to Unix. In fact, IBM had initially written proprietary windowing software for the RT, its first AIX platform. However, once XWindow was announced, IBM scratched that software and adopted XWindow.

IBM intends to put its value-added stamp on extensions to its AIX software. Plans are in the works to create bridges to SAA and the Unix environment. For example, SQL will be an applications interface in both environments that would lend itself to a common bridge. Other bridges would be created by communications protocols. IBM is quick to point out that both the Unix and the SAA environments have standardized on the C programming language.

CONCLUSION. We were very excited to note that IBM is continuing to change its tune about standards. IBM's possible acceptance of Apollo's NCS environment is significant enough to push NCS into a dominant position in the distributed networking environment.

The obvious implications of these developments as well as other indications show that we are indeed entering a new era of computing. The days of time-sharing seem to be numbered. The concept of being able to use the most appropriate computing resource in a virtual distributed environment is a reality too powerful for any vendor to ignore.

We were also pleased to see IBM making such a large and long-term commitment to the Unix operating system. We like the notion that there will be connections between Unix and SAA. This appears to be a wise move and one worth watching. In essence, IBM is hedging its bets. Although the company will continue to pour resources into its proprietary operating systems, Unix presents an opportunity to silence the argument that IBM is knee deep in too many operating systems. We predict that combining SAA and Unix will have important future ramifications for IBM's future software strategy.

—JSH

•PLEXUS•

A Windows-Based Development Tool

As a follow-on to its Extended Data Processing (XDP) system, Plexus has brought forth a toolkit to aid the developer of Mixed-Mode data processing applications. The tools, called XDE (Extended Development Environment), run under Microsoft Windows and are intended to simplify applications development in the graphically oriented XDP environment. XDE includes

forms for building menus and creating reports, and C Code for run-time modules. "XDE was intended to help the developers so that they don't have to know about handling image databases and working with windows," notes Maurizio Gianola, Plexus's software tools manager. According to Gianola, the toolkit embodies all the knowledge of windows and the SQL interface that is part of Plexus's XDP system. "To work and program with windows requires a different way of thinking," he says. He points out that the image-based windowing environment is event driven. The developer has to build a critical mass before creating an application. In addition, the toolkit includes a fourth-generation language (4GL). The language, based on the Informix database management system, is used to create the reports and queries within the XDP environment.

How does XDE differ from a traditional development environment? In a typical development environment, a programmer has to begin by writing code. With the XDE toolkit, the developer actually draws boxes representing windows and menus and then uses the 4GL to create reports and queries. What makes XDE even more interesting is that it then generates the MS-Windows code. This will be a great boon to developers of multimedia, windowing applications since complex coding is required. At present, XDE runs under DOS 3.1 and MS-Windows 1.03. In the future, Plexus intends to move to MS-Windows Version 2.0. However, this goal may require Plexus to rewrite some of its code. Eventually, we are likely to see Plexus migrating to OS/2 and the Presentation Manager.

CONCLUSION. Creating a multimedia environment based on image and a windows user interface is an excellent move. We are satisfied that Plexus is providing a set of tools that will help developers reach their goals faster. However, we would still like to see a set of these tools to help end users develop their own applications. While we realize that this is a tremendous task, we think that it would be worthwhile. Once end users begin to understand the power of such resources, they will start to ask for tools that give them more control over their work. Plexus's XDE toolkit is on the right track. ◐

—JSH

•PYRAMID•

Pyramid Ushers in New OLTP Systems

Pyramid Technologies has packaged its 9000 series mini-mainframes with Sybase's database package to create a family of high-performance transaction-processing systems. It's not quite shrink-wrapped On-Line Transaction Processing (OLTP), but almost. Users have a choice of five basic systems that range in performance from 10 transactions per second (TPS) using the TP-1 benchmark on the low-end Model 50 up to 75 TPS on the high-end Model 400. With pricing for the complete systems ranging from $165,000 to $675,000, the new Relational Distributed Transaction Processing Systems (R*TP) may offer the lowest cost-per-transaction in the industry.

SYBASE. The critical component in this system package is the Sybase software. Sybase is designed to be a relational database manager with the robust performance and function necessary in an OLTP system. Sybase, in other words, is bridging two currently disparate disciplines that have heretofore required separate software packages (at least for higher-performance OLTP). (For an initial look at Sybase, see our review, Vol. 2, No. 7, page 17.)

Briefly, Sybase's competitive distinctions are: a single-process back end, extended SQL commands, the ability to store integrity rules and procedures centrally in the data dictionary, the ability to maintain the system without requiring downtime, the ability to update multiple databases in a single transaction, and a graphics user interface that minimizes programmer time.

Sybase is designed to run well across a network, with workstation front-ends and server back-ends. Currently, the hot workstation for Sybase is Sun's. However, a Macintosh version should be out shortly (especially with Apple owning a little piece of Sybase), and the worst-kept secret in the industry is Sybase's ongoing work with Microsoft.

Pyramid is going after the back-end business with its R*TP family. The vendor, which has perhaps the most aggressive strategy for pushing Unix fully into the commercial side of the house, made some changes to its Unix operating system to enhance the OLTP capabilities of the R*TP systems.

For example, Pyramid added a facility that locks the back-end server process into only one of its up to four processors, allowing the other processors in a Pyramid configuration to be used for other things—such as a front end. (Sybase uses a requester/server architecture where the application functions can be handled separately from the data management functions. The DataServer runs the data management processes on a server—the back end. The DataToolset provides a set of window-based tools for building and running applications on either a character terminal or a bit-mapped workstation—the front end. With a multiprocessor architecture such as Pyramid's, you could have one processor running the back-end DataServer application and the other processor(s) running either front-end DataToolsets or other application programs. For performance reasons, Pyramid doesn't want the back-end server process swapping out across different processors.)

To enhance overall performance, Pyramid put context-switching into microcode. And, to avoid the delays caused by Unix's blocking on a write, Pyramid added an asynchronous I/O fix to the raw file system. None of these changes disrupt OSX's compatibility with Unix standards, however.

The result of all this should be a

high-performance OLTP system that is very attractive in price and that will be a natural for incorporation in the flexible, third-generation networks that are evolving.

We'll be taking a more detailed look at Pyramid Technology's approach to the commercial marketplace in an upcoming report. ◐ —*MDM*