

DATABASE

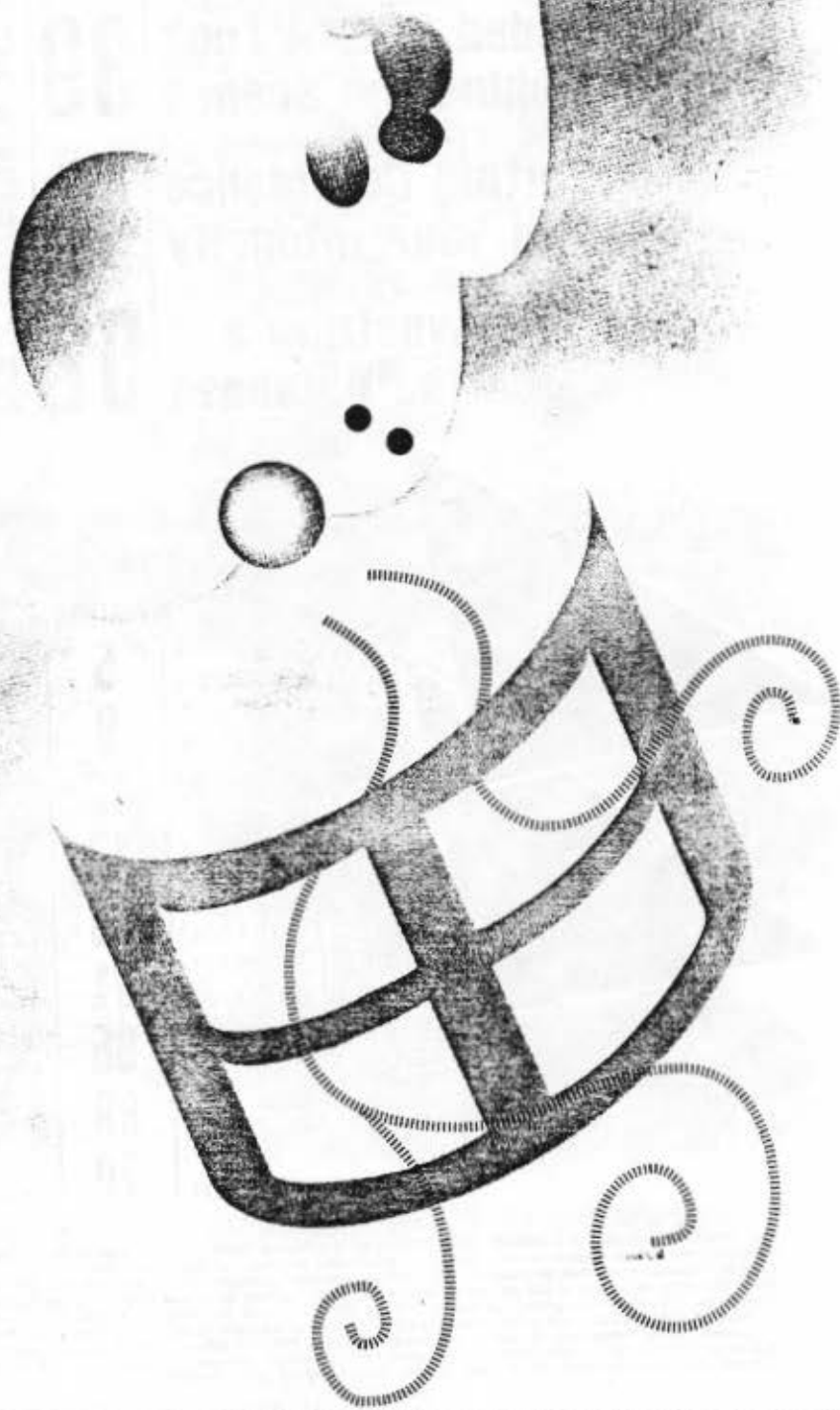
Programming & Design

**Windows Access to
the Database: Letting
Breeze Blow In**

**Putting Confidence
In Your Oracle
RDBMS Integrity**

**Embedded SQL:
What Happens
Behind the Scenes**

**Zachman's ISA
Framework: A Data
Sharing Strategy**



JUNE 1992
\$3.95
\$4.95 Canadian

DATABASE

Programming & Design

Letting the Breeze Blow In

26

DAVID MCGOVERAN

An evolutionary view helps MIS adjust to Windows-based end-user database access.

Embedded SQL: A Look Behind the Scenes

39

DAVE COHEN

Soup-to-nuts on what happens to embedded SQL in a client/server environment.

Putting Confidence In Your Integrity

46

EDWARD KOSCIUSZKO

Techniques for implementing referential and data integrity with Oracle databases.

Downsizing and the Database

52

ANAND V. RAO

The move to LAN-based operations opens the door to many new roles for the database.



DEPARTMENTS

EDITOR'S BUFFER

5

Microsoft's drama dominates DB/Expo.

ACCESS PATH

9

A reader's view on the topic of views.

DATABASE DESIGN

13

Zachman's ISA: Catalyst for better business.

DBA SHOPTALK

17

Translating DB2 for IMS technicians.

CELKO ON SQL

23

OLTP: Have to admit, it's getting better.

DESKTOP DATABASE

61

The Foxpro purchase and the fate of dBASE.

REPOSITORY REPORT

65

How attributes clear up business rules.

DATABASE LIBRARY

69

A SQL work-for beginners and experts.

PRODUCT WATCH

70

Wrapping up DB/Expo and more.

DATABASE PROGRAMMING & DESIGN (ISSN 0895-4518) is published monthly by Miller Freeman Publications, 800 Harrison St., San Francisco, CA 94107, (415) 905-2200. Please direct advertising and editorial inquiries to this address. For subscription inquiries, call (800) 289-0189 (outside U.S., (303) 447-6300). SUBSCRIPTION RATE for the U.S. is \$47 for 12 issues. Canadian/Mexican orders must be prepaid in U.S. funds with additional postage at \$6 per year. Canadian GST Permit #124513185. All other countries outside the U.S. must be prepaid in U.S. funds with additional postage at \$15 per year for surface mail or \$40 per year for air mail. POSTMASTER: Send address changes to DATABASE PROGRAMMING & DESIGN, P.O. Box 53481, Boulder, CO 80322-5481. For quickest service, call toll-free (800) 289-0189 (in Colorado or outside the U.S., (303) 447-6300). Please allow six weeks for change of address to take effect. SECOND CLASS POSTAGE paid at San Francisco, CA 94107 and at additional mailing offices. DATABASE PROGRAMMING & DESIGN is a registered trademark owned by the parent company, Miller Freeman Publications. All material published in DATABASE PROGRAMMING & DESIGN is copyrighted © 1991 by Miller Freeman Publications. All rights reserved. Reproduction of material appearing in DATABASE PROGRAMMING & DESIGN is forbidden without permission. 16mm microfilm, 35mm microfilm, 105mm microfiche and article and issue photocopies are available from University Microfilms International, 300 N. Zeeb Rd., Ann Arbor, MI 48106 (313) 761-4700.

Windows is changing some of the fundamental ways we do things in MIS.

Letting the

Here, we discuss what users should be aware of when

Breeze

researching the new marketplace

Blow

TODAY'S INFORMATION center is under siege. The popularity of personal computers, UNIX workstations, spreadsheets, graphical user interfaces, relational databases, and LANs has placed new demands on information center managers. Users expect immediate access to corporate, work group, and personal information, but do not want the specialized training that traditional access requires.

The typical Fortune 100 com-

pany maintains data in a variety of databases and file structures. Some databases reside on mainframes, while others reside on minicomputers and microcomputers. With the ever-increasing power of PCs and the variety of shrink-wrapped software now available for end users, a vast amount of information is held captive by PCs in stand-alone configurations.

Sharing information access among users presents difficult problems for information center man-

agers. Traditional data center requirements have lessened in importance; many of the assumptions that could be made regarding control of data management and information systems are no longer valid. The categories of user the center must support range from data processing professionals to end users, who generally have no exposure to, let alone training in, information systems management. It's difficult to satisfy end users and still maintain some measure of

In

control over resource management and data integrity. In this article, we will consider these issues as well as criteria for satisfying new requirements.

HISTORICAL PERSPECTIVE

Traditionally, end users were offered less, expected less, and their needs were easier to manage. When all computer hardware required special power, temperature, humidity, and other controls, it made sense to keep that expensive hardware in a single physical location. You could control computing resources, including data, in ways that would not be considered today. Similarly, the training required to operate computer systems was extensive and highly specialized. Lost computer time due to inappropriate use of facilities was intolerable and costly. Wasted CPU cycles, let alone idle MIPS, were situations to be avoided at all costs.

Initially, it was cost-effective to exert tremendous effort to optimize software for minimum resource consumption. This procedure mitigated the development of general-purpose software and made "user friendliness" less important than it might otherwise have been. With the advent of lower-cost computing power and more scalable hardware, software for end users became a possibility. The progression from special-purpose programs for each user request to today's user-development tools with graphical user interfaces has been a natural one.

Perhaps the easiest way to follow the development of the technology is to examine the need to generate business reports, an essential function for any information center. Early report generation technology consisted of using special-purpose, hand-coded programs. The programs were usually run in batch mode.

Following the availability of standard file systems and as smaller computer systems became available in the mid and late '60s, special-purpose languages such as IBM's Report Program Generator were developed. These languages reduced the amount of coding required, but did not alleviate problems associated with the program-

Traditionally, end users expected less and were offered less

development cycle. The iterative code, compile, link, run, debug, and edit cycles still consumed excessive computer power and programmer time. The basic problem with report languages is that the model used to specify the report does not match the report itself, so developers must wait for the report to run before obtaining any feedback on the correctness of their efforts.

For a while, report-generation languages improved in sophistication while remaining batch processing facilities. With the introduction in the '70s of products such as Information Builders Inc.'s Focus and Must Software International's Nomad, you could develop reports interactively, which shortened the development cycle by eliminating the compile and link steps. These products became known as query languages and were used extensively for data retrieval. Even so, the interface for these products was still the command line. Eventually, the ability to modify as well as retrieve data became a common part of query languages.

The widespread introduction of the forms metaphor for data entry and retrieval in the late '70s and early '80s led to the eventual integration of forms and query languages for report generation (IBM's Query-by-Example, Relational Technology's Query-by-Forms, and Digital Equipment Corp.'s Forms Management System are ex-

amples). The technology might be characterized as "static," in the sense that the dimensions of forms were closely tied to the physical dimensions of display terminals; graphic representations were generally not used. Even as late as 1984, commercial applications rarely made use of forms and graphics in the same package and they certainly were not integrated in the same consistent user interface.

PCs AND GRAPHICS

Of course, all of this would not have been very successful if other technology had not kept pace. The early popularity of such PCs as the Apple, Commodore, Sirius, and eventually the IBM PC made it possible to change the user interface. Low-cost graphic display and color terminals then made it possible to use the mouse input device technology invented by Doug Englebart at Stanford Research Institute in the '60s. Similarly, the work of Alan Kay at Xerox PARC led to the windows paradigm and icons.

The Smalltalk environment was the first significant realization of the use of windows and a mouse input device, but most GUI technology became widespread after Apple introduced the Macintosh in January 1984. (Apple's interest in windowing environments is not too surprising, considering that Alan Kay became an "Apple Fellow.")

On the software side, spreadsheet packages such as Visicalc and Lotus's 1-2-3 became popular, enabling users to analyze data intuitively. This ability drove the need to integrate graphic display of the results. Spreadsheet and graphics-presentation tools like 1-2-3, Microsoft Excel, and Informix Wingz have become extremely popular, especially in the fast-growing Windows environment.

Partly as a side effect of this success, information center users have come to expect access to corporate, departmental, work group, and personal data through graphical user interfaces. Since such interfaces are still not widespread in other environments, a strong motivation exists for users to access the information center through their PCs, using Windows 3.1 in particular.

CONNECTIVITY

Information centers are no longer the private domain of MIS managers, systems analysts, and COBOL programmers. We now have a more open environment in which users with little or no technical training are more numerous than data processing professionals. The physical organization of information centers has changed as well, especially with the advent of LAN technology and PC workstations. Network management is now a major concern.

Relational DBMSs are particularly useful for client/server computing, connectivity, and downsizing. Thus, in mainframe MVS environments, access to data managed by DB2 (the leading MVS-based relational DBMS) is extremely important. VAX/VMS systems, often used by departmental information centers, has become more popular in corporate MIS. Oracle, Rdb/VMS, VAX DBMS, and Ingres are the leading DBMSs on VAX/VMS platforms.

Clearly, relational DBMS connectivity is an essential component of any information center end-user tool. Three issues are key for connectivity: network support, data-exchange formats and protocols, and distribution support. Most DBMS vendors support a variety of network protocols and offer gateways to handle such problems as network protocol translation, routing, and data format conversion. In most instances, gateways run on a CPU that is distinct from either the client or server CPU, and can service multiple clients and servers. When information center data resides on a mainframe that does not behave as a server, the gateway may also convert between peer-to-peer and client/server architectures.

At the other end of the connection, a PC client may expect a file server as compared to a DBMS server. Again, either special programming or a gateway may be required to make the connection relatively transparent to the end user.

TYPES OF DISTRIBUTION

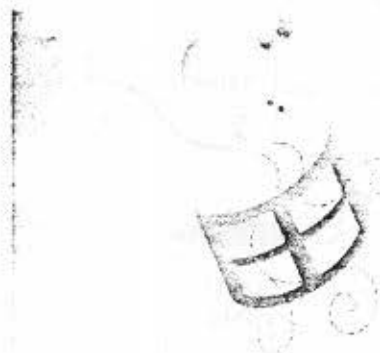
Four types of distribution services are possible: remote request, remote transaction, distributed transaction, and distributed request.

Information centers are no longer a private domain

Which one the information center chooses depends heavily upon their particular needs, but today, the most common distributed support involves remote request and remote transaction. To understand how they are used, consider the kinds of processing that are most common on workstations (or generally, PCs) connected to an information center:

□ Extract report/query processing (see Figure 1). With this read-only access to the information center's database, data is downloaded into a local database for decision support, report generation, and document preparation. Since the user defines the extract manually, accuracy and usability are key tool-selection criteria. Extract processing is very popular with information centers because control over the database's integrity is so much easier than if the user is allowed to update the database directly.

Tools for this purpose must provide an efficient and user-friendly means for extract definition. Named collections of tables (not necessarily in one database or even at one remote site) accessible to the product should be listable and user selectable. The user should not be burdened with network or remote connection issues; these issues should be handled as part of installation and configuration tasks. The data's downloaded format should also be selectable,



so that access by popular workstation tools is transparent.

□ Data entry staging (see Figure 2). Here, data is entered on the workstation, processed as much as possible, and then uploaded to the information center. Thus, the workstation becomes a staging area. Typically, the information center will provide a program for the update process and may also prevent direct update of the database by uploading processed data to temporary tables and controlling the process of merging it into the corporate database.

As with tools for extract processing, the user should not be burdened with network or remote connection issues. Ideally, data staging should produce named collections of tables to which target tables can be preassigned; the upload of this processed data to the target tables is then transparent to users, who need only confirm that the data staging operations are complete on the named collection. The processing capabilities of data staging tools can span a wide range of functionality, from simple forms data entry and editing to spreadsheets and graphics editing functions.

□ Direct update (see Figure 3). In this case, a copy of data to be updated is downloaded to a local database. Data is entered and updated locally, but these operations are reflected directly in the corporate database. This form of update processing is possible when the necessary data can be manually extracted at the beginning of a session and any updates are certain not to require transaction management. For example, the database copy of the extracted data may be read-locked (either by the database or by operational procedures) so that no more than one source of updates is possible at any time.

Direct update requires support of networks with a higher bandwidth than either extract processing or data staging, since messages and data are going to be passed from the workstation and the information center in a more conversational-like mode. It is essential that tools for direct update ensure data integrity prior to committing updates, which typically occur on a small number of rows

at a time. The number of rows that are sent to the information center should be selectable in the tool. This feature can be a very important part of transaction definition as well as a matter of network performance.

□ Browse and update (see Figure 4). This form of processing is the most difficult one to support without compromising either multiuser concurrency or data integrity. It also usually wreaks havoc the basic idea behind client/server processing, which assumes that requests to the server are atomic and not conversational. The user browses through an extract or a remote database and may edit selected rows. The problem here is that the processing takes a relatively long time in transaction management terms. If locks are held in the database, they prevent other users from making updates. If locks are not held, many types of integrity problems can occur. The so-called optimistic concurrency control method of checking updated rows for changes by another user before committing them works only if updates do not logically depend on other rows.

Tools that support browse and update processing need sophisticated control over how requests are submitted to the data-

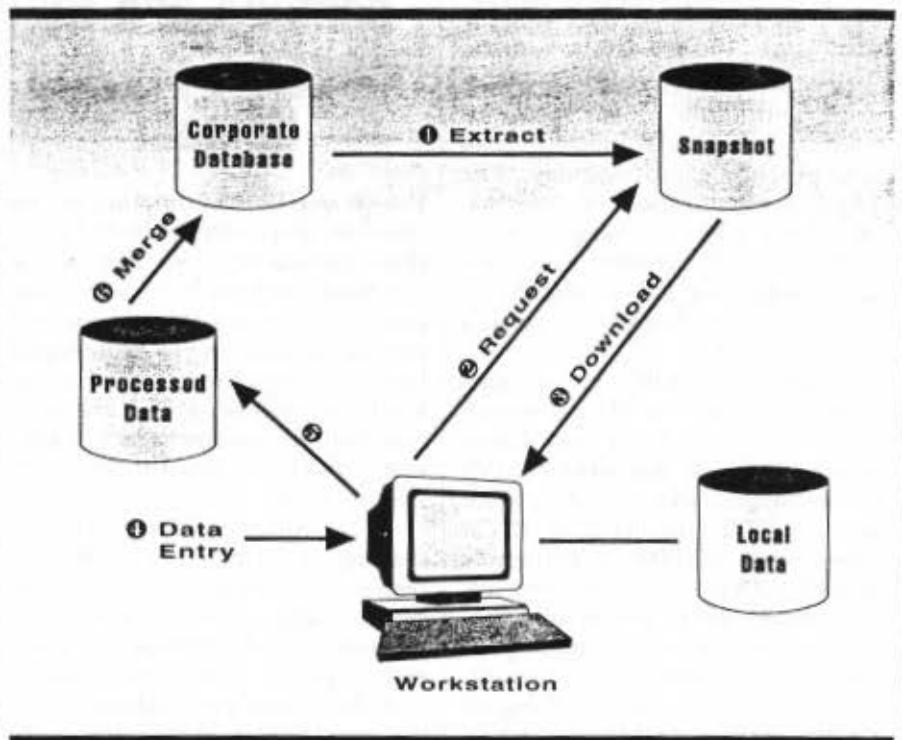


FIGURE 2. Data entry staging.

base, what locks are held and for how long, transaction isolation levels, recovery under power failure, and so on. The ability to scroll forward and backward through displayed data and to process result sets recursively becomes important. The typical user is not likely to understand the issues involved, so it is important that the

DBA be able to configure the environment appropriately and according to the requirements of each particular task.

Understanding distribution services is important. Suppose a product does not explicitly support distributed request (that is, access or update of multiple data sources in a single query while guaranteeing that all of the request will complete or none of it will). However, let's say it does provide remote request and simultaneous access to multiple data sources. The typical user cannot be expected to understand the impact. If the user is allowed to display data from multiple sources simultaneously—one window for each—this process is, in effect, an on-screen join. If the user now performs an edit in one window while examining data in a second window, all the problems of distributed transaction management must be considered. Short of either precluding an update/insert on any window when multiple (nonstatic) data sources are being used or implementing a two-phase commit protocol and appropriate distributed transaction management under the covers, there seems to be no way of preventing this problem.

While isolated instances of other forms of processing occur (in-

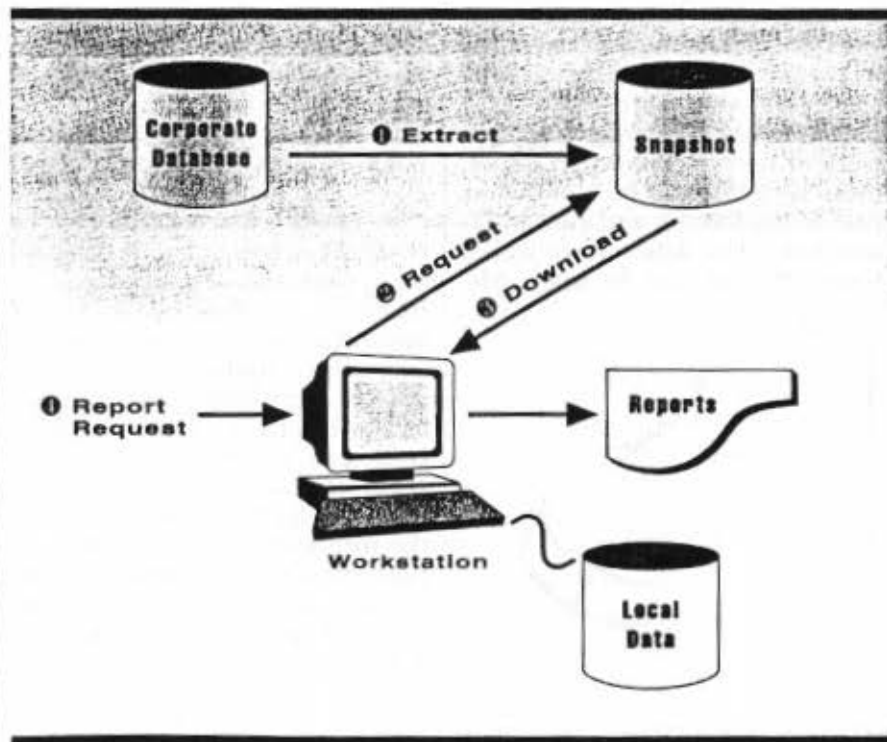


FIGURE 1. Extract report/query processing.

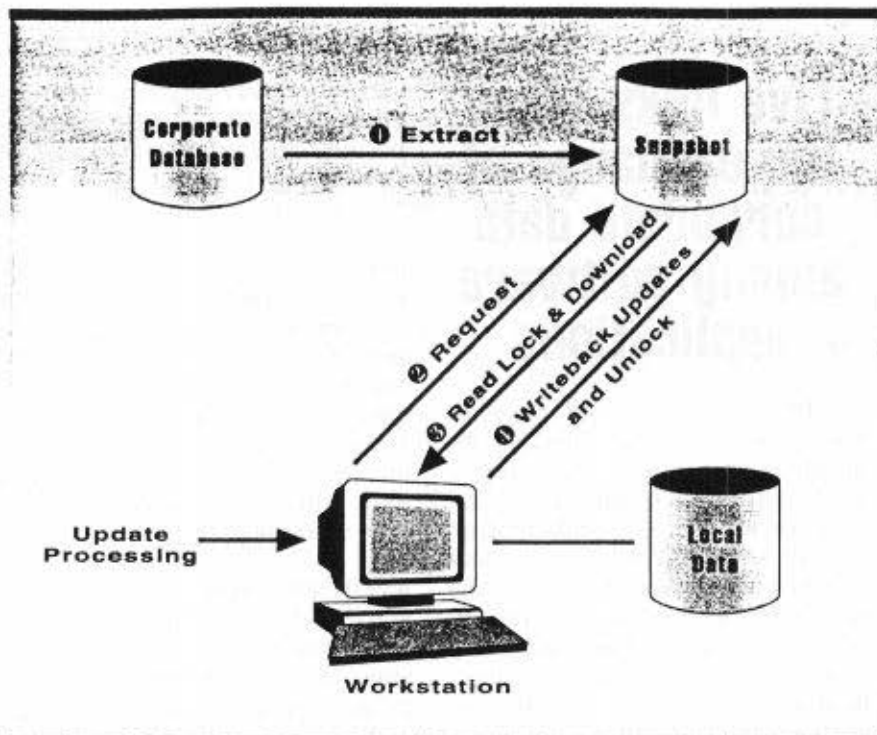


FIGURE 3. Direct update.

cluding those involving distributed transactions and distributed request), support for this type of processing is the primary requirement.

END-USER TOOLS

Key to the evolution of information centers and their support for applications is software functionality. We can categorize software as follows (note that these categories are neither exhaustive nor mutually exclusive):

- Systems administration
- Design and development facilities
- Utilities and services (including communications and database services)
- Custom applications
- Commercial applications
- End-user tools.

The category receiving the most attention today is end-user tools. Many information centers are positioned to move into client/server computing, but have had to use the older technique of extract processing while waiting for adequate end-user tools. At the PC workstation, however, we find many popular packages, such as spreadsheets, report writers, and browsers. They are often designed for use with file servers rather than database servers. (For additional information on how to choose end-user

tools, see the sidebar, "Selecting End-User Tools.")

While these packages are useful additions to the information center arsenal until better tools are available, they may also impose a burden if not used properly or if they use shared computer resources inefficiently. The tools can reduce the applications

backlog and, with a consistent, well-designed user interface, can reduce training requirements. If the tool is flexible, it can serve many purposes but require user training only once.

THE USER INTERFACE

The overwhelming success of Microsoft's Windows 3.0 has changed end-user expectations and helped solve a portion of the end-user training problem. Because all Windows programs have a consistent user interface, users no longer expect to spend long periods of time learning the computer or mastering a new program. Windows programs also have the added advantages of graphics and, most importantly, the ability to exchange data among programs without programming. Because they can connect existing tools directly, users will require fewer new applications.

How well does a Windows application support the user interface? Here are some questions to ask:

- Does the application present a comprehensible mental image or metaphor?
- How consistent is the organization of data, tasks, and functional roles?
- Is the scheme for navigating the application efficient?

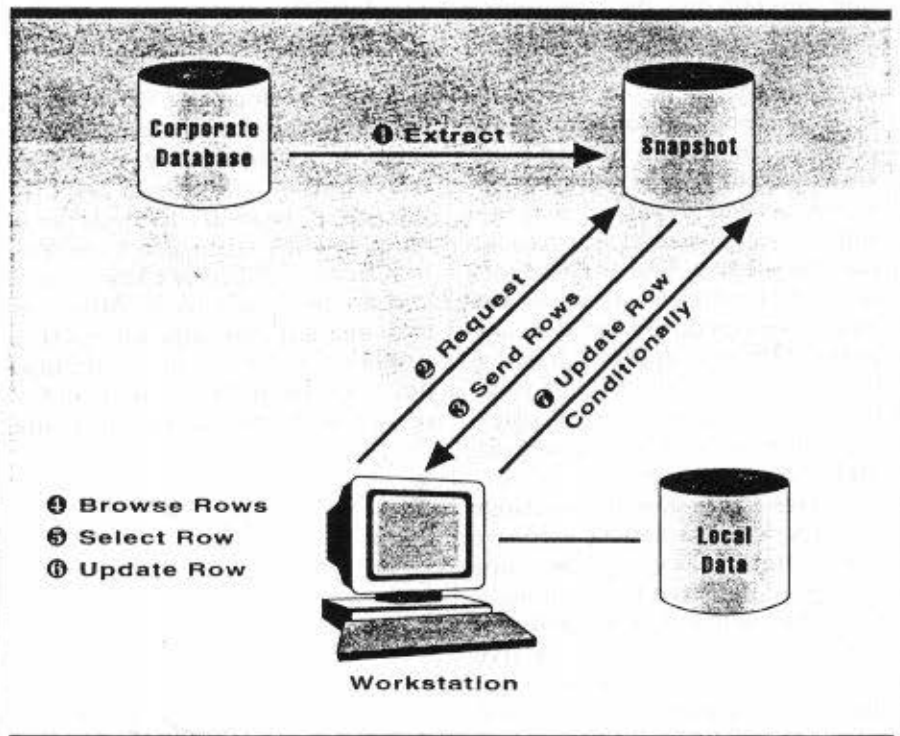


FIGURE 4. Browse and update.

Is the appearance and layout on screen appealing?

Does the product have a good "feel"? For example, are the sequence of events when interacting with the application natural?

Are multiple representations of data provided?

Is consistency used where a difference would evoke "user surprise," and is variety used to obtain the users attention?

Is the number of controls minimized?

Have legibility and readability been given adequate attention? For example, have fonts and text size been properly selected?

Have icons been selected for meaningful symbolism?

Does the product make adequate use of feedback (errors, progress indication, and so on)?

DATA EXCHANGE

Getting information from one program to another has always been difficult. Character-based applications typically specify their own data storage and display format, and conversion between formats is usually tough. With Windows, traditional conversion methods—writing data file conversion programs or creating paper printouts from one program and then reentering data into a second program with redundant data entry—are no longer necessary.

One popular solution to this problem has been to cut and paste. Unfortunately, this method involves several steps and may not support noncharacter information such as graphic data or text fonts. If the data source changes, the cut-and-paste operation must be repeated to obtain updated information. In place of cut-and-paste technology, real-time, enterprise-wide information sharing is increasingly possible using live links.

Live links allow the exchange of corporate data among software applications. Two applications connected by a live link automatically change if any of the underlying files or data change. A live link includes instructions about data access, communications, and integrity. It is triggered by a preset action by the application's user (for example, simply opening a document or calling for data

Live links allow the exchange of corporate data among software applications

might trigger it), and executes transparently to the user. For example, a user who creates a report with Gupta Technologies' Quest may include a spreadsheet produced with Excel. A live link would ensure that changes to the source spreadsheet will be reflected in the report. Essentially, live links ensure that each application is sharing up-to-date information at that same time. In a windowing environment, both applications can be active in their own window. The arduous manual process of copying data among applications is automated, so that it is no longer time-consuming and error-prone. Live links also reduce local storage requirements by replacing redundant information with pointers to shared data. This elimination of redundant data helps the user maintain a consistent view of the data, thus enhancing integrity.

Within the MS DOS environment, Dynamic Data Exchange (DDE), a protocol for interprocess communications, enables a program to establish live links and use another program's data as though it were its own. DDE is based on the idea of a conversation between a server (data source) and a client (data user). Although each request can only reference a single data object, a given application can be both a client and a server and can access multiple

servers or provide data to multiple clients. The use of DDE links can become quite complex in a multi-tasking environment.

Windows supports two forms of DDE linkage: temporary and permanent. A temporary link disappears once the information has been exchanged; a permanent link remains in effect. Permanent links can be "hot," meaning that the data is exchanged automatically as soon as it changes, or "warm," meaning that data is exchanged only when the receiving application requests it.

AVOIDING SQL

SQL is the standard query language for relational database systems. Invented at IBM in the early 1970s, SQL was designed primarily for ad hoc, interactive querying of relational databases. Because it was one of the first high-level, nonprocedural languages, and because of IBM's tremendous influence in the marketplace, it has received broad acceptance. While SQL has many flaws (such as redundant forms of expression and nonintuitive syntax), it is certainly easier to learn than C or COBOL and is used extensively by DBAs and programmers.

However, most end users do not use SQL, do not wish to use it, and do not have the four to six months it takes to become a reasonably proficient user. Most users prefer to interact with a form, table, or spreadsheet, which automatically generates the required SQL statements. From time to time it is necessary to generate more complicated SQL than can be easily and unambiguously represented with such an interface. Under these circumstances, some means of modifying the generated SQL or of expressing the request more directly is required.

MULTIPLE VIEWS

End users may find any of several views of requested data desirable—an example of multiple representations in a user interface. Three textual types of view are common:

Table view. When the user wants to browse larger amounts of data, a table view consisting of multiple rows and columns can of-

Selecting End-User Tools

WHERE DO YOU BEGIN when choosing end-user information center tools? What factors would you consider essential? With such tools, of course, the best way to select them is by examining the tools directly. The following features of support criteria are a start; the list is not meant to be exhaustive, nor is it meant to be used as a score sheet.

■ **DOS.** As the operating system with the largest installed base and number of products, DOS support is key. For some applications, however, the ability to move transparently to IBM's OS/2 Presentation Manager could be important.

■ **Microsoft Windows 3.1.** Windows' popularity, many benefits, and reportedly fluid migration path to OS/2 PM make support here crucial.

■ **Effective GUI use.** The tool should exploit the GUI's richness. The presentation should be visually attractive, with an intuitive functional meaning of icons. The tool shouldn't present too much information at one time. This feature is especially important for timid—perhaps first-time—users and for sales demonstrations.

■ **User-friendly data manipulation.** To minimize training, the tool's functions should be intuitive and pleasurable to use. For more complex functionality, beginner and expert modes should be supported.

■ **Error reporting.** This feature must be as friendly as the rest of the interface, and must be interpreted in a manner appropriate to the data source.

■ **Clipboard.** For the serious data analyst, the ability to run multiple query sessions and cut-and-paste among them using the Windows clipboard offers important productivity benefits.

■ **Dynamic data linkage.** The ability to link multiple data sources to a single document, which is then updated dynamically as the sources are updated (called live-link capability) is

powerful. You can provide familiar, spreadsheet-style update activity, yet maintain a more modular and loose coupling among data sources and targets.

■ **Multidatabase access.** Many corporations use multiple databases and would like to access them from a PC workstation-based tool. Thus, the tool should be able to access databases managed by DB2, Oracle, OS/2 Extended Services Data Manager, SQL Server, and other major products. Within the tool's functionality, SQL access to these DBMSs should be complete. A more extensive capability—merging data from multiple sources into a single table or report—is not yet offered by many products. This powerful feature would free organizations from the limit of accessing only one database at a time.

■ **Clear position on distributed services.** Users shouldn't be concerned with distributed transaction management. If difficult problems here can't be solved completely, the services should be avoided. The vendor should make a clear statement about the kind of distributed support the tool provides and the kinds of processing permitted.

■ **Uniform data source.** The tool shouldn't force users to import data and convert data formats prior to using a particular data source in any given view: table, form, or report.

■ **Tables.** The table view of data must be supported and should include "spreadsheetlike" editing capabilities.

■ **Forms.** With integrated forms, users can change data views. The best capability today is Query-by-Forms.

■ **Extensible applications development.** It should be possible to save, copy, and rename reports, queries, and forms, and to extend their functionality with more sophisticated development tools.

■ **Easy-to-use report writer.** A WYSIWYG preview function is par-

ticularly useful in a report writer. With font control, graphics, and image capability, a report writer can approach desktop publishing capabilities.

■ **Ability to create local tables.** This function is crucial to development and testing; most DBAs would not want to allow users uncontrolled manipulation of a DBMS. In a stand-alone environment, some means of creating the database and its data must exist. The user shouldn't have to step outside the product to perform this function.

■ **Database creation and storage.** The tool should let users name, type, and size data that must be stored, as well as restructure databases. The tool should maintain objects such as reports, queries, forms, and so forth in a database, rather than in a proprietary local file format.

■ **Set processing.** To get the right rows, users shouldn't be forced to import a table a row at a time. If the tool supports direct update processing, the point at which updates to the database occur should be controllable. For example, automatic update, insert, or delete on leaving a row in the display should be provided as well as deferred update of multiple rows. Control of update modes will allow tuning for concurrency, network traffic, and database performance.

■ **Standard SQL.** The product should support either ANSI standard SQL or a commonly used dialect, such as DB2 SQL. Ill-defined, ambiguous, or proprietary definitions of the types of SQL syntax and behaviors supported inevitably leads to inconsistencies among various modules and the behavior vis-a-vis the databases.

■ **Complete SQL support.** Although SQL should not be visible to the casual user, it should be fully supported. Products are often weak in the area of data and transaction definition.

—by David McGovern

fer more utility. A table view lets users edit, delete, and insert data in multiple columns and rows, a behavior familiar to spreadsheet users.

Form view. When the user

wants to view a single row at a time, possibly with a large number of columns, form representation can be helpful. Query-by-Forms may be the best choice, letting the user enter example data values to

access data that best matches the example.

Report view. A report view cannot be used to edit data. Its primary purpose is the presentation of data, generally in printed form.

The degree of sophistication of a report view can vary from a simple columnar report format to complex reports with control break (group) processing and computations.

FOUR EXAMPLE PRODUCTS

As stated earlier, few end-user tools currently support information center access. But by looking at four prominent products positioned to meet this need, we can explore differentiating features and clarify the challenges.

□ **Borland International Inc.'s Paradox SQL Link.** Borland's Paradox was designed for data entry, query-by-example, and reporting. Applications can be built using Paradox Application Language (PAL). The Paradox Engine is designed for efficient use of a (proprietary) flat-file database and file servers. Paradox has a good user interface, and it is easy to learn and use. The product has been retrofitted with SQL access capabilities through Paradox SQL Link, which supports information center access, primarily through extract processing. Paradox SQL Link imports data directly from either Microsoft or Sybase SQL Server and converts it into Paradox file format locally.

Several problems arise with this approach. One is the added performance cost of downloading and uploading data across the network, and of converting it between database and Paradox formats. Perhaps more important, transaction management using this technique can be costly and frustrating to the user. Because multiple copies of data exist, a change to the Paradox copy must be checked for consistency. The idea is to reread the source data row following a change, compare the originally read row to the new copy, and apply the update only if they match.

This technique is common among front-end tools that support "browse and update." Unfortunately, it has costs both in terms of efficiency, storage requirements, and integrity. Obviously the operation requires multiple reads of the same data and requires that the edits be buffered locally until they are checked. Most important, this technique assumes that each row is being modified indepen-

Few end-user tools currently support information center access

dently. If the user bases a modification on the displayed values of other rows, possibly in other tables or other windows, data integrity can be compromised since these other rows are not checked when the update is applied.

If the checks fail, the user must redo the work, leading to considerable frustration. In some applications, these considerations are not significant and can be overlooked. When the amount of data involved becomes large or transactions become even moderately complex, the technique is undesirable. Of course, this problem does not manifest when Paradox is used without SQL Link and is not as likely to become serious until used in the information center environment.

□ **Software Publishing Corp.'s InfoAlliance.** InfoAlliance is characterized as a data source integrator. It provides direct access to live data, the approach most often used by client/server products. InfoAlliance provides a forms design, data entry, report generation, and supports image and document scanning. A significant feature is the ability to scan a hard copy form and then create an InfoAlliance form on top of it, with automatically computed and formatted fields. It contains its own database engine, which is used to integrate data from multiple foreign databases.

InfoAlliance runs under OS/2 Presentation Manager and Windows. The product is intended to simulate distributed transaction management across multiple data sources while hiding the database location from the user (database tables are presented to the user as familiar PC files).

InfoAlliance is a valiant attempt to make client/server relational DBMSs behave like a file system. The product is extremely useful for users who have no de-

sire to understand transactions, data distribution, or SQL, especially in environments in which relational DBMSs are not being used for mission-critical processing or where data integrity problems can be ignored.

□ **Oracle Corp.'s Oracle Card for Windows.** Oracle Card is a graphical, HyperCard-like product introduced originally for the Macintosh, but is available now for Windows. It uses a stack of index cards as a metaphor for applications (hence its name). Oracle's success with relational DBMS products in the minicomputer market is certain to make Card successful; it is designed as a front-end tool for Oracle version 5 and above.

The Card stack is a collection of cards that makes up the particular application. Cards contain objects (information) that can be in either the background or foreground. Objects in the background are shared among cards in the stack. Objects in the foreground are specific to the card.

Oracle Card consists of a Stack Builder for building stacks, a Query Builder for building simple on-screen and text reports, and a Table Builder for creating and managing tables and views. Reports can have a header and a footer, but are essentially columnar. Applications built using Table Builder can have up to eight database tables on a card; master-detail and master-detail-subdetail applications are supported. A Toolbox provides facilities for drawing and painting objects, creating fields, obtaining object information, browsing a stack, and switching between background and foreground.

Oracle Card works best for users who have some understanding of building and using small applications. Unlike SQL*Forms, Oracle Card is not intended for high-end, industrial-strength applications, nor is the product's focus information access: Its strong points are graphical display and manipulation features. Oracle Card's HyperCard-like quality is very different from the forms orientation of SQL*Forms; applications developed in one are not portable to the other. This approach poses problems for Oracle Card users as they move from one tool to the other,

which they may do because the application requirements have become more demanding.

□ Gupta Technologies' Quest.

Quest can be used for the distribution services such as online data access, data extract and upload, local processing, and data entry staging. How the services are used is up to the information center manager. Likewise, database security is managed by the DBA. Quest performs a limited presentation format very well, with little effort on the user's part. Like Paradox, it is easy to learn and use.

Quest has four parts: Tables, Query, Report, and Catalog Manager. Table provides both a table and a form view of data, along with the ability to format, edit, and manipulate data. Query is the user interface used to develop information center requests and generate SQL. It focuses on SQL capabilities while hiding the SQL terminology. Once defined, queries can be saved for reuse. Report is a banded report generator with which the user can develop custom reports and define groups (such as control breaks), headers, and footers. Quest reports are compatible with Gupta's ReportWindows report generator. With the Catalog Manager, users view and manage data definitions in the database. Using live links, Quest can be used to drive spreadsheet and graphic analysis products; thus, users can select tools and integrate them for information center access.

Quest was designed for SQL relational databases to be used by users with little or no development experience. It does not perform distributed transaction management; these functions are left to the DBMS engine. A database engine for local data management is bundled with the product. Like other products, features to automate and control the uploading and downloading processes are not supported in the current version.

CHALLENGES

Supporting today's information centers requires tools for the workstation and the database server or host platform. Control over integrity, security, and availability

of corporate data must remain with the information center. At the same time, workstation functionality must be exploited to free the user from the constraints of traditional data access methods. Tools today have some of these required features; however, we should expect to see some rapid improvements over the next couple of years as vendors gain experience with information center needs. Information center control, user-friendliness, and end-user productivity should not be too much to

expect from one product. ■

The first version of this article, "Windows SQL Access for Today's Information Centers," was originally published as a white paper by Gupta Technologies.

REFERENCES

McGovern, D. and C. J. White. "Clarifying Client/Server," *DBMS*, 3(12), November 1990.

White, C. J. "Client/Server Computing with DB2," Gupta Technologies, March 1990.

David McGovern is the president of Alternative Technologies, a consulting firm that provides relational DBMS applications design and development services.

Desktop DBA™

Serving up database servers... Windows style!



Desktop DBA delivers Windows® 3.0 front end power to match that of SQL database servers. With Desktop DBA, you can manage any number of database servers on your network simultaneously, each in its own window.

Desktop DBA goes beyond even the high end utilities available in mainframe environments, with features like drag-and-drop database migration. Point-and-click object management. Automatic corruption detection and repair.

All of this means you don't have to accept any more excuses about how client/server technology is lacking tools for DBAs and developers. With Desktop DBA, all the pieces are in place.

Desktop DBA for Microsoft® SQL Server™ and SYBASE® available now. Desktop DBA for Oracle® available early 1992.

DATURA The front end company

1926 East Parham Road
Richmond, VA 23228
804 264-1225
FAX 804 264-1297

CIRCLE 15 ON READER SERVICE CARD