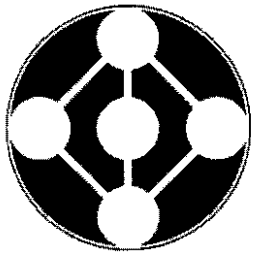

Alternative Technologies

Quest
Windows SQL Access
for
Today's Information Centers

prepared for

Mr. Umang Gupta
Gupta Technologies
1040 Marsh Road
Menlo Park, California 94025
415/321-9500

by

David McGoveran
Alternative Technologies
150 Felker Street, Suite E
Santa Cruz, California 95060
408/425-1859

April 23, 1991

UNCLASSIFIED
FOR OPEN DISTRIBUTION BY GUPTA TECHNOLOGIES

CONTENTS

Today's Information Centers

Today's information center is under siege. The popularity of personal computers, spreadsheets, graphical user interfaces, relational databases, and LANs has placed new demands on information center managers. Information center users expect immediate access to corporate, work group, personal information, but do not want the specialized training which traditional access requires.

The typical Fortune 100 company maintains data in a variety of databases and file structures. Some of those databases reside on mainframes while others reside on minicomputers and microcomputers. With the ever increasing power of personal computers and the variety of shrink wrapped software now available for end users, vast amounts of information is held captive by personal computers in stand alone configurations.

Sharing information access among users presents difficult problems for information center managers. On the one hand, traditional data center requirements have lessened in importance. At the same time, many of the assumptions that could be made regarding control of data management and information systems are no longer valid. The categories of user which must be supported range from data processing professionals to end users generally having no exposure (let alone training) in information systems management. This later category of user is difficult to satisfy while simultaneously maintaining some measure of control over resource management and data integrity. In this paper the issues involved will be explored and criteria for satisfying the requirements presented. While the state of the art does not completely address these difficult problems, products have begun to appear which focus on the issues.

A Historical Perspective

Traditionally, end users were offered less, expected less, and their needs were easier to manage. In the days 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. This made it possible to 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 -- specific to a particular computer system. The cost of computing was such that lost computer time due to inappropriate use of the facilities could not be tolerated. The idea of wasted CPU cycles, let alone idle MIPS, was 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 against 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 hardware that was more environment tolerant, software for end users became a possibility. The progression from special purpose programs for each end user request to today's end 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 programs coded by hand. These report 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 1960s, special purpose languages such a RPG were developed. <<Fig. RPG sample program>> These languages reduced the amount of coding required of the programmer, but did not alleviate the problems associated with the program development cycle. The iterative code, compile, link, run, debug, and edit cycle could still consume 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 the developer must wait for the report to run before obtaining any feedback on the correctness of his/her efforts.

For a while, languages for report generation improved in sophistication while remaining batch processing facilities. With the introduction in the 1970s of commercial products like Focus and Nomad it became possible to develop reports interactively shortening 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 << Focus Fig.>>. 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 (such as IBM's QBE -- Query-by-Example, Relational Technology's QBF -- Query-by-Forms <<Fig.??>>, and Digital Equipment Corporation's Forms Management System) in the late 1970s and early 1980s led to the eventual integration of forms and query languages for report generation. The technology might be characterized as static in the sense the dimensions of forms were closely tied to the physical dimensions of display terminals and 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 almost certainly were not integrated in the same consistent user interface.

Of course, all of this would not have been very successful if other technology had not kept pace. The early popularity of personal computers such as the Apple, Commodore, Sirius, and eventually the IBM PC made it possible to change the user interface. The introduction of low cost graphic display and color terminals made it possible to use the mouse input device technology invented by Doug Englebart at Stanford Research

Institute in the 1960s. Similarly, the work of Alan Kay at Xerox PARC led to the windows paradigm and to the use of icons. The first significant realization of the use of windows and a mouse input device was the Smalltalk environment.

On the commercial side, the popularity of spreadsheet packages such as Visicalc <<Fig?>> and Lotus 1-2-3 provided end users with the ability to perform data analysis in an intuitive manner and drove the need to integrate graphic display of the results. Spreadsheet and graphics presentation tools like Lotus 1-2-3, Microsoft Excel, and Informix WingZ have become extremely popular, especially in the Windows environment.

Most of this technology first found widespread use in the Apple Macintosh personal computer, introduced in January of 1984 after an ill-fated and earlier attempt with a system known as Lisa. Considering that Alan Kay became an "Apple Fellow", this interest by Apple Computers in windowing environments is not too surprising.

Microsoft Corporation released Windows 1.01 for MSDOS in November of 1985. <<Fig. >> From that time until the release of Windows 3.0 in May of 1990, Windows enjoyed only partial success. Today it is the fastest growing user interface in the industry and marks the clear success of graphical user interfaces. 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 than personal computer environments, there is a strong motivation for end users to access the information center through their personal computers and using Windows 3.0 in particular.

Gupta Technologies was one of the early participants in the Windows market with SQLWindows, along with such products as Aldus PageMaker and Microsoft PowerPoint for Windows.

Market Forces Generate Requirements

Traditionally information centers were closed, tightly controlled environments. Physically they were centrally located, not only to improve control of the environment, but also because the required equipment was large and costly. They were also more difficult to operate. <<See Figure 1.>>

The 1980's saw a number of changes in the information center market. Several key factors characterize these changes:

- IBM DB2 legitimization of mainframe relational DBMS
- increased use of DEC VAX systems
- client/server computing

- increased interest in connectivity
- the trends towards interoperability, downsizing, and open systems

At the same time, information center end users have begun to demand certain functionality. These key factors in this area have been:

- demand for PC-based rather than terminal based applications
- demand for Windows applications in particular
- interest in sharing data and accessing corporate data
- a refusal to learn and use language interfaces like SQL

Information centers are no longer the private domain of MIS managers, systems analysts, and COBOL programmers. A more open environment exists in which users with little or not technical training are more likely to be present than are data processing professional. The physical organization of information centers has changed as well, especially with the advent of LAN technology and PC workstations making network management a key concern of information center managers.

Computer operations problems persist, but the level of sophistication has greatly improved. Security administration is no longer a question of physical access, but is usually handled through software, as are many other tasks that were once manual. Perhaps the most important factor in the support and evolution of information centers is the functionality of the software available. Such software can be categorized as follows:

- System Administration
- Design and Development Facilities
- Utilities and Services (including communications and database services)
- Custom Applications
- Commercial Applications
- End User Tools

End User Tools for Information Center Access

These categories are, of course, not exhaustive nor are they necessarily mutually exclusive: a particular program may belong to more than one category. Of these categories the one receiving the most attention today is end user tools. End user tools which

support information center requirements are difficult to find, a lack which has slowed the introduction of client/server computing.

Commercial end user software packages can be a desirable addition to the information center arsenal. They can also impose a burden if not properly used or if they use shared computer resources inefficiently. The main benefit of such tools is that they reduce the backlog of applications which must be designed, developed, and maintained. If they use a well designed user interface, they can reduce the amount of training required. One flexible tool gets used for many purposes but requires training the user community only once. Similarly, a consistent user interface across applications reduces the training time for multiple applications.

Relational DBMSs

The use of relational DBMSs is particularly important in that it enables the client/server computing, connectivity, and downsizing potential of information centers. In the MVS mainframe environment, access to DB2 data is extremely important. According to a recent study by Sentry Market Research, the MVS mainframe database environment is dominated by DB2 at 26% and continues to erode the IMS share currently at 20%. Oracle accounts for another 4% of this market.

Another dominant information center force is the VAX/VMS market. VAX/VMS systems are often used by the departmental information center and are increasingly used by corporate MIS departments. Part of this trend is due to the enhanced capabilities of the maturing VMS operating system and part of it is due to the price/performance curve offered by VAX systems with an ever widening range of computing power. In this market, Rdb/VMS dominates with 19%, VAX DBMS with 13.8%, Oracle with 18.2%, and Ingres with 12.9%.

These factors make it clear that RDBMS connectivity is an essential component of any information center end user tool. Support for access to DB2, Rdb/VMS, and Oracle are all important. For the moment, non-relational data remains a factor but is being replaced by relational systems.

Of course not all database systems are currently connected in client/server architectures. For example, a recent study of Fortune 1000 sites by Business Research Group of Newton, Mass., found that 14% used Oracle and 13% used DB2 in client/server systems but that less than 8% used Rdb/VMS, Sybase SQL Server, Microsoft SQL Server, or Ingres. The lack of front end tools compatible with information center needs seems to be responsible. Nonetheless, revenues in the front end tool market has been estimated at increasing at 30%-40% annually indicating that progress is being made.

Connectivity

As noted above, connectivity has become increasingly important in the information center. There are three issues involved in connectivity: network support, data exchange formats and protocols, and distribution support. The approach being taken by most DBMS vendors is to support a variety of network protocols and to offer gateways to handle such problems as network protocol translation, routing, and data format conversion. In most instances gateways run on a CPU which is distinct from either the client CPU or the server CPU and can service multiple clients and multiple 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 Services

The type of distribution services required depend heavily on the particular installation. there are four types of distribution services possible: remote request, remote transaction, distributed transaction, and distributed request. Today the most common forms of distributed support involve remote request and remote transaction. To understand the use of these, consider the kinds of processing that are most common in information center connected PC workstations.

- *Extract report/query processing*

  Information center data is downloaded into a local database for decision support, report generation, and document preparation. This kind of processing involves read only access to the information centers database. The extract is defined manually by the user, so that the kind of tool provided for the purpose can have an impact on accuracy and usability. Control over the integrity of the database is therefore easier with extract processing than if the end user is allowed to update the database, making extract processing very popular.

- *Data entry staging*

  Data entry is performed on the PC workstation, processed as much as possible, and then uploaded to the information center. Thus, the PC workstation becomes a staging area. Typically information center provides a program for the update process and may prevent direct update of the database. Instead the processed data is uploaded to temporary tables and then merged into the database under information center control.

- *Direct update*

  Data entry and update operations are performed on a local copy of the data and reflected directly in the 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.

- *Browse and update*

  This is the most difficult form of processing to support without compromising either multiuser concurrency or data integrity. The end 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.

<<< Figure x. Common Distribution Services >>>

Understanding these types of distribution services is important. For example, suppose a product does not explicitly support *distributed request* (i.e., 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) but does provide remote request and simultaneous access to multiple data sources. The typical end-user can not be expected to understand the situation. If the user is allowed to simultaneously display data from multiple sources, one window for each, this is effectively 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 need to be considered. There seems to be no way of preventing this problem short of either (1) precluding an update/insert on any window when multiple (non-static) data sources are being used, or (2) implementing a two-phase commit protocol and appropriate distributed transaction management under the covers.

While isolated instances of other forms of processing do occur (including those involving distributed transactions and distributed request), support for these is the primary requirement.

Security Issues

User authorizations to access data in an information center must be managed at the database level just as access to a system is normally managed at the operating system level so that it can not be subverted. Generally, access to shared data is managed by system and database administrators. If end user tools also provide local user account and password protection, it is important that the tool integrate well with the security system of the information center. The most common approach to security is to pass user identification and password information on the server for validation. This allows security to be centrally managed, while not disabling that ability to control access to local data at the workstation level.

The User Interface

The overwhelming success of DOS Windows 3.0 has changed end user expectations. Users no longer expect to spend long periods of time learning how to use the computer or mastering a new program. All Windows programs have the same fundamental look and feel, a consistent user interface. This ease of use and reduced training are compelling reasons for using a GUI. Other advantages include the ability to deal with graphics applications and the ability to exchange data between programs without programming. This later advantage can significantly reduce end user requests for new applications by allowing the end user to connect existing tools directly.

A particular implementation of a Windows application should be evaluated for how well it supports the user interface. It is not sufficient to be a Windows GUI application. Here are some additional 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 through the application efficient?
- Is the appearance and layout on screen appealing?
- Does the product have a good "feel", i.e., 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 variety used to obtain the users attention?
- Is the number of controls minimized?

- Have legibility and readability been given adequate attention, i.e. 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, etc.)?

Live Links

One area in which Windows applications can contribute to both information center management and use is data exchange. Getting information from one program to another has always been a difficult problem. Character-based applications typically specify their own unique format for data storage and display. The traditional method of writing data file conversion programs and paper printouts with redundant data entry is no longer necessary. Conversion between formats is usually difficult.

One modern approach to satisfying this problem is cut-and-paste. Unfortunately, it involves several steps and may not support non-character information such as the text font used or graphic data. If the data source changes, the cut-and-paste operation must be repeated to obtain updated information. Instead real-time, enterprise wide information sharing is increasingly possible using live links.

Live links provide a means of exchanging corporate data between 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 off the application user, whether it is simply opening a document or calling for data, and is executed in a manner that is transparent to the user. For example, a user who creates a report with Quest may include a spreadsheet produced with Excel. A live link would ensure that changes to the source spreadsheet would 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 and manual process of copying data between 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 DOS environment, a protocol for interprocess communication called DDE (Dynamic Data Exchange) was invented by the Microsoft Excel development team. DDE provides a program with the ability to use another program's data as though it were

its own and to establish live links. It 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 multitasking environment. A predefined format is used for exchanging text string data called clipboard format. Formats for other types of data are application specific.

Two forms of DDE link are supported by Windows: temporary and permanent. A temporary link disappears once the information has been exchanged while a permanent link remains in effect. Permanent links can be either hot, in which case data is exchanged automatically as soon as it changes, or warm, in which case it is exchanged only when the receiving application requests it.

Avoiding SQL

The standard query language for relational database systems is SQL (Structured Query Language). Invented by IBM in the early 1970's, SQL was designed primarily for ad hoc interactive querying of relational databases. Because it was an early example of a high-level non-procedural language and because of the influence of IBM on the market place, it has been widely accepted. As languages go, it is certainly easier to learn than a 3GL programming language such as C or COBOL. Unfortunately, the language has many flaws, redundant forms of expression, and is often non-intuitive.

Most end users do not wish to learn SQL and do not have the four to six months it takes to become reasonably proficient in the language. While its non-procedural character helps users express simple requests without needing to understand how the request will be satisfied, SQL requests rapidly become cumbersome and difficult to understand when they are used with multiple database tables. Users tend to make subtle errors that even SQL experts do not spot quickly. For these reasons, most users prefer to interact with a form, table, or spreadsheet which automatically generates the required SQL. 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 more directly expressing the request is required.

Multiple Views of Data

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 browsing larger amounts of data, a table view consisting of multiple rows and columns can offer more utility. A table view has a behavior that is more familiar to users of spreadsheets, letting users edit, delete, and insert data in multiple columns and rows. <<Quest Fig.>>

- *Form View* - When viewing a single row at a time, possibly with a large number of columns, a forms representation can be helpful. Query-by-Forms capability may then be important, allowing the user to enter example data values and access data which matches the example. <<Quest Fig.>>

- *Report View* - A report view can not 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. <<Quest Fig.>>

Gupta Technologies' Quest

Quest is designed to solve the problem of end user access to the information center. It can be used for extract and upload of data, for local processing, and as a staging area for data entry operations. Rather than attempt to provide every possible presentation format and force the end user to learn complex commands, it performs a limited presentation format very well and with little effort on the part of the user.

Quest differs from and is not designed to be an Executive Information System (EIS). For example, the graphical presentation and analysis of data is a common feature of an EIS which is not directly supported in Quest. Nonetheless, through live links it can be used to drive spreadsheet and graphic analysis products like Microsoft Excel. This feature provides the potential for the Quest user to select the best (or their favorite) tool for such purposes and *integrate* information center access.

There are four parts to Quest: 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 generates SQL. It is focused on SQL capabilities while hiding SQL terminology. Once defined queries can be saved for reuse. Report is a banded report generator which allows the user to develop custom reports, define groups (i.e., control breaks), headers, and footers. Quest reports are

fully compatible with ReportWindows. The Catalog Manager is used to view and manage data definitions in the database.

Competitive Products

Few products exist today which are designed to meet the needs of end user information center access. Two approaches to the problem seem to dominate. The first approach is one of supporting remote access with extract processing. This is the approach taken, for example, with products designed for use with file servers such as Borland's Paradox. Remote data is accessible from information center resources using Paradox SQL Link. Paradox SQL Link is used to import data directly from either Microsoft or Sybase SQL Server and convert it into Paradox file format locally.

There are several problems with this approach. The added performance cost of downloading and uploading data across the network, and of converting it between database format and Paradox format is one factor to be considered. Perhaps more important, transaction management with the technique can be costly and frustrating to the user. Because multiple copies of the 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 which 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 independently. 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.

The second approach to information center support is to act as a data source integrator using live data. This is the approach taken most often by client/server products such as Software Publishing Corporation's InfoAlliance. There are several differentiators between the products. InfoAlliance runs under OS/2 Presentation Manager (Windows support has been announced). InfoAlliance is intended to simulate distributed transaction management across multiple data sources while hiding the database location from the user. It supports IBM Extended Edition Database Manager, dBASE, FoxPro, and Clipper files with support

for IBM's DB2 and Microsoft SQL Server expected sometime in 1991.

The Future

Quest is designed to be integrated with other products rather than competing with them. The technology will continue to develop and improve as feedback from users is obtained. For example, a forthcoming product called OpenQuest will allow third party developer to integrate Quest capabilities with their products by providing an API. Later versions of Quest can be expected to support graphical objects. Also, a multiuser version of Quest called Quest PowerPak will enable up to five Quest users on a LAN using SQLBase or using routers to existing databases. Express Windows will be included in Quest PowerPak.

An Overview of Selection Criteria

The following is a list of criteria for selecting an end user information center tool. The list is not meant to be exhaustive. Rather it is a start, designed to help information center managers in developing their own selection criteria. Where appropriate comments regarding Gupta Technologies' Quest product have been included. The reader is encouraged to examine Quest directly, and especially where such comments are missing.

DOS support - As noted above, the large installed base of DOS users and the large number of DOS products demand continued support for DOS.

Gupta Technologies products run under DOS as compared to other database client/server products which require OS/2.

Effective GUI Use - The tool should make use of richness of the GUI.

Microsoft Windows support - Microsoft Windows is extremely popular. It provides many benefits and an easy migration path for users that will ultimately use Presentation Manager.

Quest, like most other Gupta Technologies products, is a Windows application.

Dynamic data linkage - The ability to link data sources to a single document which is then dynamically updated as the sources are updated (the *live link capability*) is very powerful. It provides the kind of spreadsheet-style update action users are familiar with, yet maintains a more modular and loose coupling between data sources and targets.

Quest provides live links to products such as Express Windows, Microsoft Excel, and Microsoft Word for Windows. <<Quest Fig.>>

Clipboard - For the serious data analyst, the ability to run

multiple query sessions and cut-and-paste between them using the Windows clipboard offers important productivity benefits.

Quest supports the Windows clipboard.

Visually attractive - The presentation of Quest is visually attractive and, with a few exceptions, does not appear to present too much information to the user at one time. This is important for timid, perhaps first-time, users and for sales demonstrations.

Intuitive icons - The meanings of the icons used for the various functions of the product should be intuitive.

Quest provides icons which are easy to recognize, and supplements them with text labels.

Error reporting - Error reporting needs to be as friendly as the rest of the interface, and must be interpreted in a manner appropriate to the data source.

User friendly data manipulation - The functionality obtainable from the product must be user friendly and intuitive so that training is minimized and use of the product is perceived as pleasurable. If complex functionality is included, both beginner and expert modes should be supported.

Quest provides access to data with a few carefully selected presentation formats and without supporting complex application design. It concentrates on minimizing the effort and knowledge required of the user.

Tables - The table view of data must be supported and should include "spreadsheet-like" editing capabilities.

Quest uses the table or spreadsheet paradigm as the primary view of data for browsing, editing, analysis, etc.

Forms - An integrated forms capability allows the user to change views of the data. Preferably this should include Query-by-Forms capability.

The Quest Forms View provides single record at a time browsing capability.

Report writer - The report writer should be easy-to-use though without limiting the user. A WYSIWYG preview is particularly useful. With font control and graphics and image capability, a report writer can approach desktop publishing capabilities as they might be interpreted for report writing.

The Quest Report View provides a banded report generator and a WYSIWYG view of each band. It uses an outline paradigm for scripts which is novel and potentially productive. It allows the user to specify a border for an object. The size of the border

need not be specified but is sized automatically to surround the data. User-defined computations are supported.

Database Creation - There should be an intuitive way to name, data type, size, etc. data that needs to be stored and to restructure databases.

The Quest Catalog Manager allows the user to create and modify tables, and to *zoom in* on a table or column for more detailed information (i.e., columns in a table or characteristics of a column). The Catalog Manager will allow users to examine the definition of a view. <<Quest Fig.>>

Set-processing support - The user should not have to import a table row at a time in order to find the rows desired. If direct update processing is supported, 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.

Quest supports set-processing.

Standard SQL - The support of SQL should be either the ANSI standard or some common dialect such as DB2 SQL. An ill-defined, ambiguous, or proprietary definition of what SQL syntax and behavior is supported will inevitably lead to inconsistencies between the various modules and the behavior with various 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 definition and transaction definition.

Multi-database product access - The ability to access database products such as DB2, Oracle, Extended Edition Data Manager, SQL Server, etc. is essential. Many corporate environments have multiple database products in use and need to access them from a PC workstation tool.

Quest supports the ability to access a DB2, Oracle, and SQLBase. Support for additional database products is planned.

Multiple data sources - The ability to access data from multiple data sources and merge them into a single table or report is powerful and is not offered by many competitive products. Considering the fact that most corporations have multiple databases installed, data access can not be limited to a single database at a time. <<Gateway Fig. ??>>

Quest supports access to multiple data sources.

Uniform data source support - The user should not be

constrained to import data and convert data formats prior to using a particular data source in any given view: table, form, or report.

Quest handles transparent access to data, regardless of the view used.

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

Quest has been designed so that reports, and queries can be saved and reused. Reports are compatible with ReportWindows. If a report is not realizable within Quest or if the complexity of a report grows significantly over time, the report can be modified with ReportWindows.

Clear position on distributed services - The user should not have to be concerned with distributed transaction management. Particularly difficult problems in this area should be avoided if they can not be completely solved. A clear statement of the kind of distributed support provided by the tool and the kinds of processing permitted should be made by the vendor.

Quest is intended to be used primarily for data entry staging and for extract report/query processing. Other forms of processing are supported as well.

Ability to create local tables - The ability to create local tables is required for development and testing. Most DBAs would not want uncontrolled manipulation of a DBMS while a naive end-user used Quest to access shared data. In a stand-alone environment, there must be some means of creating the database and its data. The user should not have to step outside Quest (or its level of sophistication) to perform this task.

Quest comes with and fully supports single-user DBWindows, a Windows version of the SQLBase relational DBMS.

Complete support of foreign RDBMS - Within the context of the functionality of the tool, SQL access of foreign RDBMSs should be complete.

While some limitations do exist to SQL access of foreign relational DBMSs, Quest minimizes the differences within the context of the tool.

Storage in a database - Objects such as reports, queries, forms, etc. should be maintained in a database rather than in a proprietary local file format.

Today, Quest maintains objects in local file format.

OS/2 potential - The ability to move transparently into an

OS/2 and Presentation Manager environment can be important for certain applications.

A Presentation Manager version of Quest is in development.

Other Gupta Technologies Products

Gupta Technologies offers an extensive line of client/server products to support the information center including SQLBase, SQLWindows, SQLTalk, ReportWindows, ExpressWindows, etc. Gateways to Oracle and DB2 are available. For more information, contact Gupta Technologies.

For Further Reading

D. McGoveran and C.J.White, "Clarifying Client/Server", DBMS, Nov. 1990, Vol.3, No.12, pp.78-90

<<others?>>