Coexistence Strategies

Integrating Existing Systems with Open Systems Technologies

By Eric Wasiolek

ORACLE®

Contents

Preface

➡ The Coexistence Problem

Many organizations are interested in moving to open systems, particularly UNIX systems, because of their excellent price/performance. At the same time, these organizations have substantial investments in proprietary systems which they want to maintain. Large volumes of data and many applications still reside on these systems, and a significant investment in MIS staff training has been made. The challenge for many organizations, is to forge a strategy which will allow them to take advantage of new, open systems technology, without losing their investment in proprietary systems. They must forge a strategy that allows both their proprietary systems and new open systems to work together and to be fully utilized. Such a strategy is called "co-existence." This strategy must be implemented with little or no disruption to the existing enterprise.

This paper will examine the issue of coexistence in today's organization:

- What is a consistency solution?
- What benefits does a consistency solution offer?
- What is the nature of proprietary and open systems?
- What is required to integrate proprietary and open systems?
- What products does Oracle offer to implement a coexistence solution?

Introduction

➡ What is a Coexistence Solution?

A coexistence solution creates a computing environment where open systems may work in conjunction with proprietary systems. Various approaches may be used to allow open systems, to work in conjunction with existing applications, data, and computers. New fourth generation language applications development tools can be used to quickly build applications that work in all computer environments, using the graphical user interfaces and existing databases in those environments. Systems can be networked together to share data and applications. Existing applications needn't be discarded; rather, they can be used to access data in new relational databases that are installed in the environment. A coexistence solution provides the best of both worlds: using new open system technology in conjunction with existing applications, systems, and databases.

➡ Benefits of Coexistence

Coexistence strategy allows you to:

- Fully utilize your existing investment in proprietary systems.
- Use newer and cheaper technology and reduce the total cost of your computing environment.
- Have more flexibility in choosing future hardware and software purchases.
- Easier to accommodate organizationa change.

Protecting Your Existing Investment

A coexistence solution allows an organization to continue to use existing applications and databases on proprietary systems in concert with newly developed applications and databases on open systems. An MIS staff is able to continue to use their expertise on proprietary systems, and gradually gain expertise on open systems technology.

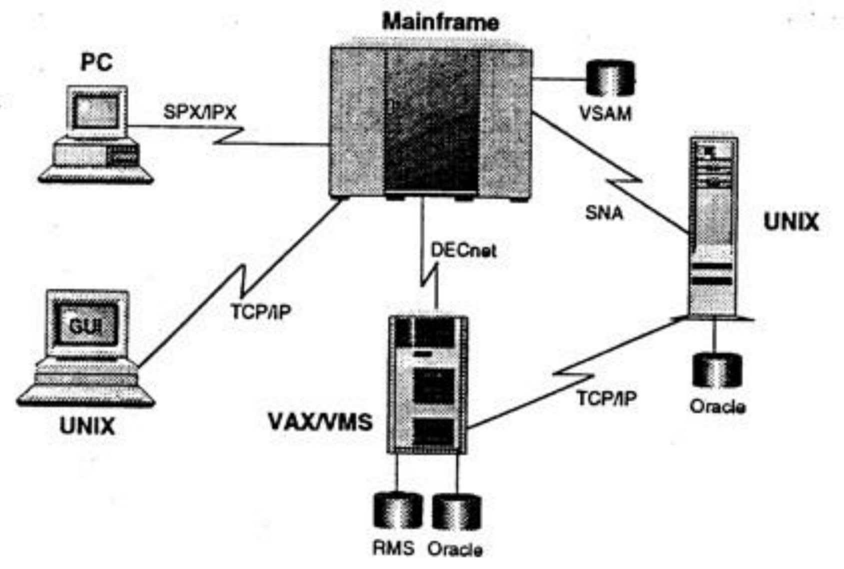

FIGURE 1 *A Coexistence Solution* is one where multiple diverse systems, applications, and database management systems work together to provide users, developers, and administrators a cooperative computing environment.

Use Newer and Cheaper Technology

A coexistence strategy allows new open systems to be purchased, along with applications and databases that run on these systems. Moving to open systems means the ability to take advantage of newer and cheaper technology. Because similar open systems are offered by many vendors, the multivendor competition drives prices down.

By being able to incorporate the less expensive open systems into an information management solution, the total cost of the solution decreases. For example, an organization may currently run small departmental applications on the mainframe, where processing time is expensive. By acquiring and connecting open systems to the mainframe, those same applications could be offloaded to run on workstations.

Flexibility of Choice

Coexistence allows more flexibility in choosing the best technology to fit the information system requirement. When users have a choice, their applications requirements are not constrained by the technology. Instead, they may define what type of system is needed to run their application, and then the system can be selected to meet that requirement. For example, the inventory management division of a manufacturing organization may want to display multiple windows with color graphs of inventory levels during the manufacturing process, but the current inventory system in the data center supports only block-mode terminals. A flexible solution would be one where the manufacturing group may purchase a UNIX workstation, with a colorful graphical user interface, connect this to the mainframe, and have the workstation display in bar chart form the data stored on the mainframe. In this case, technology meets the user's requirements. Data is located on the most appropriate platform regardless of where the application is running, and the end user selects the most suitable interface.

Accommodating Organizational Change

As organizations grow and change, it is imperative to be able to grow and change the software and hardware configuration of the computer environment. When major organizational changes occur, such as the acquisition of another organization, different technologies can be introduced to the corporate information system very quickly. Newly acquired systems will seldom be the same as existing systems. Access to data and applications will need to be accommodated with as little disruption as possible. A coexistence strategy allows existing and acquired systems to work together effectively.

We will now briefly review the different nature of open and proprietary systems, which a coexistence solution integrates.

➠ Proprietary Systems

'Proprietary' systems are systems where the hardware, operating system, and system software, all come from one vendor. These systems are optimized to solve certain types of applications like computationally intensive tasks, or handling many simultaneous transactions.

Because proprietary system hardware and software come from a single vendor it is difficult to port applications between these systems and other systems. It is difficult to share data with other systems.

In many respects, the short-comings of proprietary systems outweigh the benefits. This fact poses a strong argument to move information processing solutions towards open systems. Nonetheless, proprietary systems are still excellent at processing certain types of tasks, and the investment in them requires that they continue to be used as a part of the overall solution.

➡ UNIX and Open Systems

Open systems, in contrast, are systems based upon international standards where the components may be provided by different vendors.

Open systems are ones which have: application portability, scalability, interoperability, and are based on international standards.

Application Portability

On an open system such as UNIX it is relatively easy to run the same application on computers provided by different vendors, with little or no modification to the code. An application can be developed on a system of one type, say an system from by Hewlett-Packard, and run on a system of another type, say an NCR tower. Because both systems run UNIX, the application execution environment between the two systems is similar enough that few or no changes to the application need be made for it to run in either environment.

Scalability

Open systems allow the same application to run on systems of different sizes, potentially from different vendors, with little or no modification. By 'different size' systems is meant systems with different architectures, different numbers of CPU's, amounts of memory, and disk space. This feature of open system software is called 'scalability.' As an example, the UNIX operating system can run on machines as small as PCs and workstations, and as large as mainframes, and supercomputers. (See Figure 2)

Interoperability

Interoperability' means that one application can easily exchange information with another application. The two applications could be on the same or different computers. This 'exchange' of information allows each system to use the facilities of the other system, to cooperatively process a task.

Standards

Standards specify interfaces that allow components of systems provided by different vendors to interoperate. Standards thereby allow customers to select components of their systems from different vendors, and yet have those systems work together.

The prime example of an 'open system' is UNIX. With UNIX, the same operating system software runs on a variety of hardware architectures from a variety of vendors. Because only 10% of the UNIX operating system kernal is written in machine language [assembler], the operating system is highly portable across hardware architectures. The remaining 90% of the UNIX operating system is written in the highly portable third generation language C. For this reason, UNIX is found on the greatest variety of hardware architectures of any one operating system. This fact gives UNIX a tremendous advantage: it allows applications to run on a large variety of hardware architectures with little or no modification.

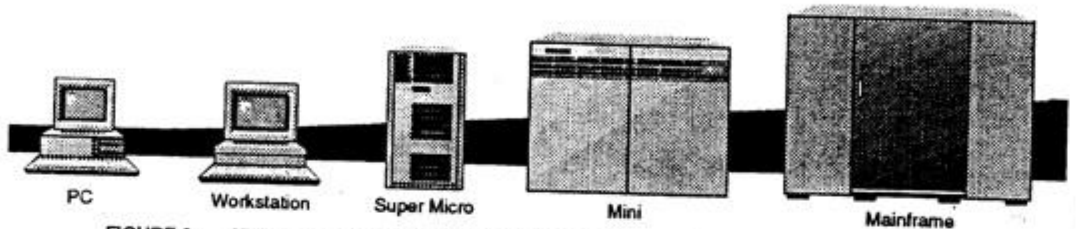

FIGURE 2 UNIX is a scalable operating system which runs on systems as small as PC's and as large as mainframes.

Approaches to a Coexistence Solution

In this section, we will discuss various approaches to implementing a coexistence solution. The approaches discussed may be used together in any combination to yield a co-existence solution.

There are five major approaches to data and application sharing:

- Upload/Download
- Application Portability
- Client-Server Interoperability
- Open Gateways
- Cooperative Server Database

➠ Upload/Download

On the simplest level, proprietary and open systems should be able to share data through an upload/download procedure. Specified subsets of data should be able to be extracted from one database and loaded into the database of another system. In this way data may be exchanged between open and proprietary systems. For example, an organization may have a wealth of data stored in VSAM files on a mainfrâme, and may want to offload some portion of that data into a relational database on a UNIX system. In this way, both the mainframe and the UNIX systems may be used for data processing.

Tools should be available to extract the proprietary data into an intermediary text format, and then download the text representation into tables of a relational database. Conversely, data should be able to be discretely extracted from the relational tables and uploaded into the file format of the proprietary system.

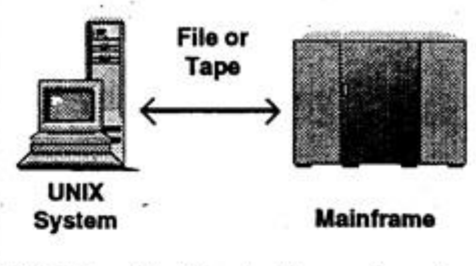

FIGURE 3 Upload/Download allows systems of different types and sizes to share information through periodic manual transfer.

➠ Application Portability

A second approach is to develop applications that can run on both open and proprietary systems. By deploying the same applications users can easily switch between systems without needing to be retrained. Common applications in both environments creates a unified computing environment for users.

Different computer systems not only run different operating systems, but they also use different graphical user interfaces, and different data management systems. Therefore, for applications to be portable in mixed machine environments, applications development tools must be available to allow applications to be written once, and without modification, be able to run across different operating systems, to use the user interface native to a particular operating system, and to access data in different data management systems.

Portability Across Graphical User Interfaces

Users will want to run the application under a user interface that is familiar to them. For example, if an application is developed and run on a VMS system, as a character input-output application, and then ported to a MacIntosh, the MacIntosh users will not be satisfied to run the application on their Mac as a character input-output application. Mac users will want to be able to utilize the MAC user interface with which they are familiar.

Developers must have tools which allow them to re-execute their same application under different user interfaces, without having to modify their source code.

This sort of portability is possible, as long as the application makes calls to a layer of software which in turn calls the graphical user interface, i.e., the application does not call the user interface directly. In this manner, the application is not tied to a particular graphical user interface, but rather can merely be recompiled or re-executed to use a new graphical user interface.

Portability Across Data Management Systems

Once an application is moved to a new operating environment, it may be that the data management system in that environment is different than the data management system the application originally used.

To make it easy to run the same application against different data management systems, tools are required that allow applications to be portable across data sources as well. Consider an employee resource application that runs on a VMS system, using an RMS file system. Suppose that a new division wants to run the same application, but wants to store the employee records in a relational database on a UNIX system. The task, therefore, would involve not only porting the employee resource application from VMS to UNIX, i.e., across operating systems, but also from an RMS data interface to a relational database interface, i.e., across data management systems. Portability across data management systems requires that the application's interface to the data source be consistent.

In summary, one way to unify users of diverse computers is to provide the same applications to these users regardless of their computing environment. To be able to do that efficiently, applications development tools must be available that allow applications to be developed once, and then merely recompiled or re-executed to run on a new operating system, a different user interface, accessing a different data management system.

➡ Client-Server Interoperability

A third approach to coexistence is to connect diverse systems together on a network, and utilize software that allows applications on one system to access data on another system over the network. This approach allows diverse systems to work together to cooperatively process tasks for users. Certain tasks, like application processing, could be offloaded from overburdened mainframes locating the application on machines which have sophisticated user interfaces.

Client-server computing is a key approach to co-existence. The client may be located on an open system and the server on a proprietary system, or conversely. This allows the two systems to cooperate to solve a common task. In this way the open system and proprietary system are both fully utilized, and hence, co-exist.

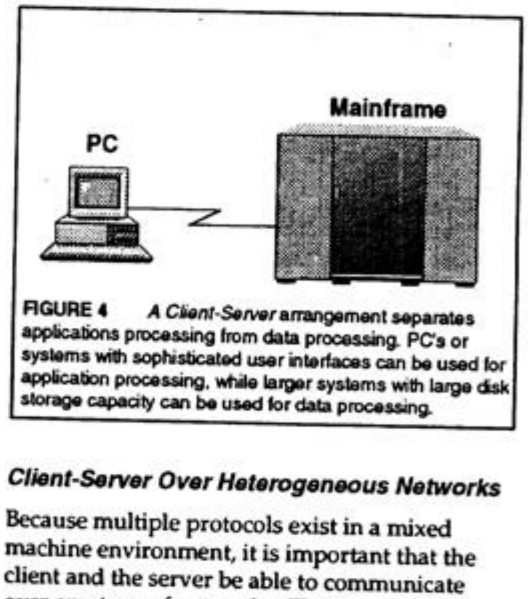

FIGURE 4 *A Client-Server* arrangement separates applications processing from data processing. PC's or systems with sophisticated user interfaces can be used for application processing, while larger systems with large disk storage capacity can be used for data processing.

Client-Server Over Heterogeneous Networks

Because multiple protocols exist in a mixed machine environment, it is important that the client and the server be able to communicate over any type of network. This is enabled by providing the client and the server a network-independent interface to the network. This interface should be the same to the client or the server regardless of the underlying protocol, and only is internally written to a specific protocol.

Client-Server Over Multi-protocol Networks

In some cases, clients will be located on different types of networks than the database servers.

Special networking software is required to allow clients and servers to exchange messages over such multi-protocol networks.

As an example, (See Figure 5) consider a client application on a UNIX system connected to a network running the TCP/IP protocol which needs to exchange messages with a server application on a MVS system connected to an SNA network, running the LU6.2 protocol. Further suppose that the Ethernet network and the SNA network are connected through a gateway. For the client and server to communicate, some service, which we will call the 'multi-protocol interchange' needs to translate the client message in the TCP/IP packet format to a client message in the SNA LU6.2 data stream format, in order for the server to be able to accept the message. In response, the 'multi-protocol interchange' service needs to translate the server response message from a SNA data stream format to a TCP/IP packet message format in order to be read by the client application.

➡ Open Gateways

Another approach is to develop new applications that are able to access existing data in a database on the proprietary system. For example, a large amount of data may reside in VSAM files on a mainframe. This approach allows new applications to be developed, with fourth generation language technology, and have those applications access data in the VSAM files.

The 'Open Gateways' approach has many advantages:

- **Data is left where it is** — it is not necessary to move any data from the existing data management system. The familiar data management system may continue to be used.
- **Existing applications continue to run against the database** - existing applications, in our example, VSAM applications, continue to run against the database, even while new applications are being developed. This way users are not disrupted from their normal tasks.
- **New fourth generation language applications can access the existing data** - great productivity gains can be achieved by utilizing 4GL technology to develop new applications against the existing data.

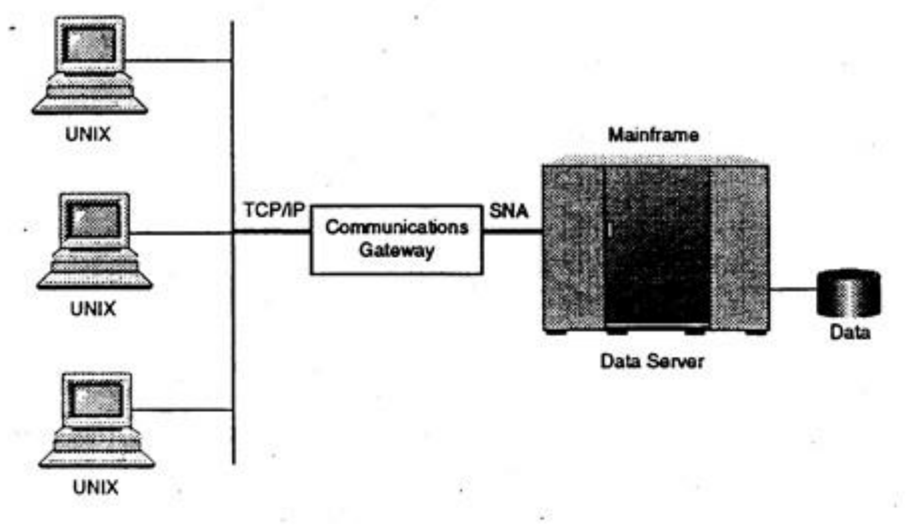

FIGURE 5 Client-Server operation over a multi-protocol gateway for clients and servers to interoperate.

- **Applications can reside on a different system than the database** - the new applications can be developed and deployed on a different system than where the data resides. 4GL applications may be deployed on a UNIX system connected over a network to the mainframe. In this way newly acquired open systems may cooperatively process tasks with existing proprietary systems.

Open Gateways access existing data, and existing applications, like VSAM applications, to access data in relational databases. There are four types of Open Gateways:

- Gateways that allow SQL applications to access data in various vendors' relational databases and in non-relational databases;
- Gateways that allow existing applications, like VSAM applications, to access relational database data;
- Transparent Gateway Toolkits that allow a customer to build their own gateway, and
- Procedural Gateways that provide access to data through a procedural interface.

Gateway Type 1 - SQL Access to Heterogeneous Data

Often, there is much existing data on the proprietary system that users want to access from a new application. For example, an organization may have much data stored in VSAM files on an IBM mainframe, yet want users located on UNIX systems to be able to access that information through SQL-based applications. By using a 'client-server' arrangement, the organization can run its applications on UNIX clients and have these clients still access data on the IBM mainframe. This type of access is technically possible, but requires special technology known as an 'Open Gateway.'

An Open Gateway translates SQL calls into the native file input-output calls of the target data source. For example, an SQL statement that selects VSAM data is translated into a specific set of VSAM file input-output calls that select the desired items from the VSAM files. Open Gateways are applications that allow client applications to access data in a variety of relational and non-relational data sources.

Gateway Type 2 - Allowing Existing Applications to Access New [Relational] Data

The second problem an Open Gateway solves is to allow existing applications, such as VSAM applications, to access data in relational databases. MIS can maintain its existing investment in proprietary applications, and simply use these applications to access data in relational databases on open systems. This allows existing applications to access the same data accessed by new applications built with 4GL technology. This is a form of coexistence where the client remains on the proprietary system, but the data is stored on an open system.

Open Gateway technology is again required so that VSAM file input-output calls are translated into the semantically equivalent SQL calls to select the target data set from the relational tables.

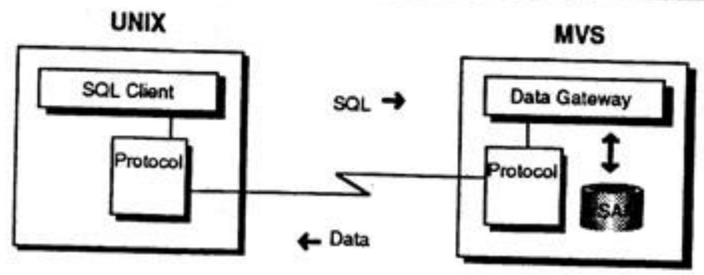

FIGURE 6 Access to existing data from standard SQL applications through *Open Gateways.*

Gateway Type 3 - Open Gateway Toolkits

Because there is such a diversity of data management systems, it is important to have technology that allows an organization to build its own gateway to proprietary file systems and data management systems. The technology must minimize the amount of needed programming while allowing adequate access to the data.

Gateway Type 4 - Access to Existing Procedural Data

Not all data is stored in databases. A large amount of the world's data is stored in applications that have a procedural interface. Data is also stored in databases which have procedural interfaces, like transaction processing systems like CICS, IMS DC, TUXEDO, and DB2 Stored Plans. A successful coexistence strategy allows access to data through a procedural interface. This is required whether the access is to the an existing transaction processing system or to user applications, e.g., C routines.

➠ Cooperative Server Database

A final approach to 'coexistence' may be called 'Cooperative Server Database.'

A Cooperative Server Database allows multiple data sources, say those in proprietary file structures and those in relational databases to be logically combined. A Cooperative Server Database allows a single user request to access data in multiple databases simultaneously. These databases may be located on both proprietary and open systems.

The advantages to this approach are:

- **Data may be located on the machine where it is most frequently used** - now users do not need to be connected over remote communications link to a centralized databank on a mainframe. Instead, data can be moved to the systems where users most frequently access it, and users can still maintain transparent access to all data.
- **Data may be redistributed without affecting user's access** - as an organization's makeup changes, data needs to be re-distributed to new sites: offloaded from mainframes or transferred to newly acquired systems. A Cooperative Server Database insures that wherever the data is moved to users continue to have transparent access to it, as if it resided on their local system.
- **DBMSs may be tied together into a cooperative computing environment** - information residing at multiple sites can be automatically correlated and processed as easily as if it resided at a single site. Multiple diverse systems are used simultaneously to process user requests.

There are two types of Cooperative Server Databases, homogeneous and heterogeneous .

Homogeneous Cooperative Server Database

A 'homogeneous Cooperative Server Database' is a logical database composed of multiple databases all of the same type: all instances of the ORACLE RDBMS. The homogeneous Cooperative Server data manager should allow a single SQL request to access data in both databases simultaneously. With

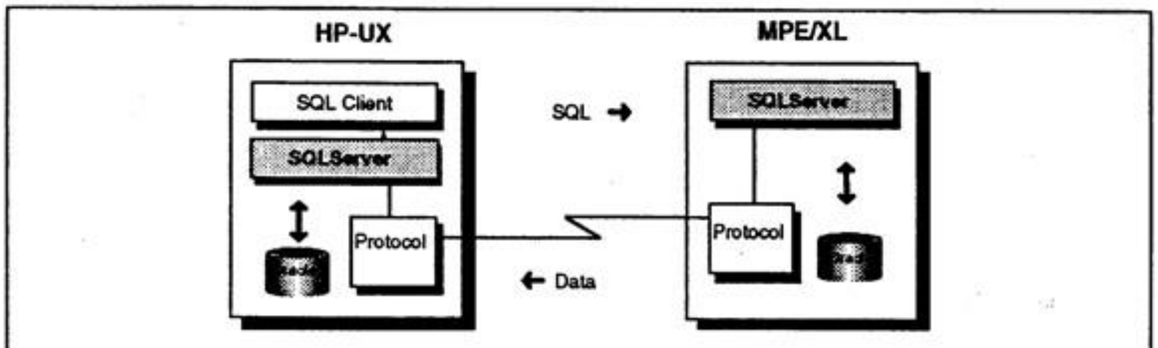

FIGURE 7 A *Homogenous Cooperative Server Database* is a logical database composed of databases all of one type.

the aid of technology called a 'distributed query optimizer' the access to those multiple databases can be optimized. The 'distributed query optimizer' determines the best order in which to send the multiple queries, and the best computer sites at which to join the data. Furthermore, a technique called 'automatic two-phase commit' allows data at multiple computer sites, potentially both proprietary and open systems, to be simultaneously updated. In the first 'phase' the Cooperative Server data manager checks to see whether all participating databases are ready to be updated. If they all respond that they are, the Cooperative Server data manager proceeds to the second phase and commits the data at multiple sites. If one or more databases are not ready to be updated, the Cooperative Server data manager aborts the transaction.

A Heterogeneous Cooperative Server Database

A 'heterogeneous Cooperative Server Database' is a logical database composed of data sources of different types. For example, a logical database composed of VSAM files on MVS and an Oracle relational database on UNIX would be a 'heterogeneous Cooperative Server Database.' Heterogeneous Cooperative Server Databases are important for 'co-existence' because they allow existing non-relational data sources to be logically combined with the new relational databases typically found on open systems.

Multiple approaches were discussed which may be used to allow data and applications to be shared across open and proprietary systems.

The table below summarizes these approaches to a coexistence strategy:

Upload/Download	Manual transfer of data
Application Portability	Across operating systems Graphical user interfaces Data management systems
Client-Server	Across heterogeneous networks Across an internetwork
Open Gateways	New applications access existing data Access to procedural servers Existing applications access relational data
Cooperative Server Database	Homogeneous Heterogeneous

FIGURE 9 Summary of Coexistence Approaches

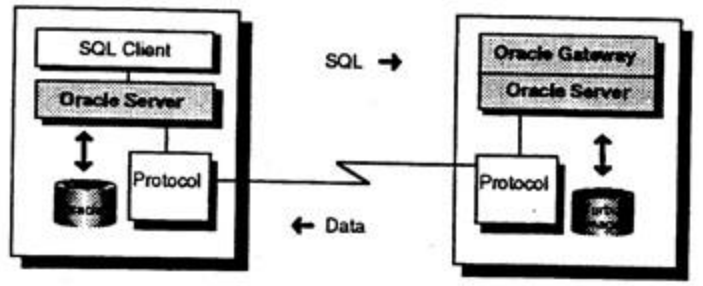

FIGURE 8 A *Heterogenous Cooperative Server Database* is a logical database composed of databases of different types.

ORACLE'S Coexistence Solution

In the last section, we discussed the requirements and various types of coexistence. It was stated that a coexistence solution:

- should provide the ability to upload and download data between disparate systems,
- the solution should allow applications to be easily ported across operating systems, graphical user interfaces, and data management systems,
- it should allow applications to be deployed in a client-server configuration for new applications to access existing data, and existing applications to access new relational data,
- data should be accessible through a procedural interface and finally
- a coexistence solution should allow a Cooperative Server Database capability to logically combine data sources at different sites.

In this section the Oracle products that make these coexistence solutions possible will be discussed.

This section will discuss the following Oracle products and product concepts:

- SQL*Loader
- Oracle Application Portability
- Oracle Client-Server: SQL*Net
- Oracle Open Gateways
- Oracle Cooperative Server Database

This section will explain how each of these Oracle products and capabilities help implement coexistence solutions.

➡ Upload/Download with SQL*Loader

Oracle can share data with any system through an upload/download scenario. This means that data can be extracted from the desired data source into an intermediary text format, and then loaded into Oracle relational database tables. This data extract and download capability is supported by a product called 'SQL*Loader' from the following databases to the Oracle database: DB2, SQL/DS, IMS, ADABAS, DATACOM, IDMS, M204, VSAM, Rdb, RMS, INGRES, Informix, TANDEM, and TURBOIMAGE, and several others.

➡ The Portability of Oracle Applications

Applications should be able to be written once, and without modification, run on different operating systems, graphical user interfaces, and data management systems. Oracle applications development tools meet this requirement better than any toolset in the industry.

Oracle applications development tools are supported on over 80 different operating system environments, including most proprietary as well as UNIX platforms. Applications using Oracle can be executed without code modifications on any of the 80 supported operating system environments. By implementing the same software solutions across open and proprietary systems, these systems are unified for users, applications developers, and system managers Both the applications which are built and the application development tools, as well as the system administration tools are portable across proprietary and open operating systems.

This portability for Oracle products and for applications written using Oracle application development tools extends to independence across operating systems, graphical user interfaces, and many different data management systems.

Portability Across Graphical User Interfaces

Oracle provides technology called the 'Adaptable User Interface' to allow portability across graphical user interfaces. For example, an application can be developed once, then simply re-executed using a different runtime to take advantage of a different graphical user interface. Oracle supports Motif and OpenLook in the UNIX domain, as well as the following graphical user interfaces on non-UNIX systems: Presentation Manager, Windows 3.0, Macintosh, DECwindows, character mode terminals, and block mode terminals.

Support for these interfaces is provided by the Oracle Adaptable User Interface (AUI) technology (see Figure 10), which maps a set of interface functionality to each GUI. Applications built

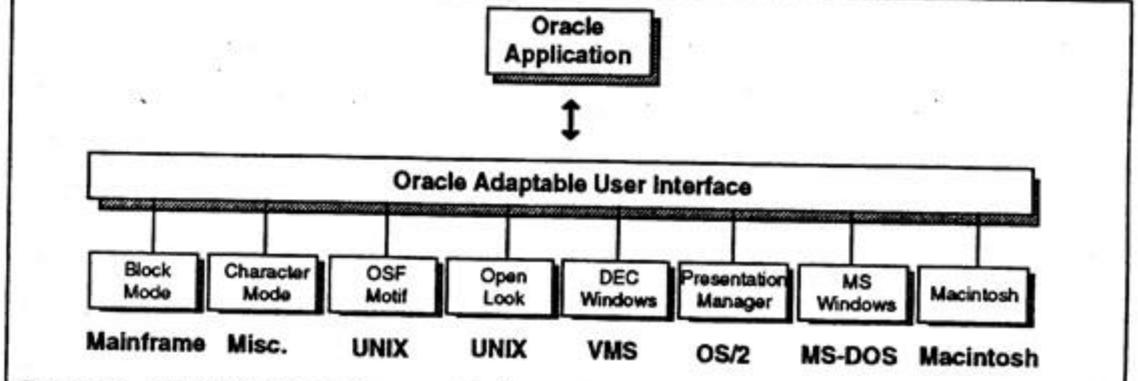

FIGURE 10 Oracle applications can be re-executed with no code rewriting to take advantage of the user interface native to an operating system. Through the "Adaptable User Interface" Oracle applications are portable across graphical user interfaces.

using Oracle tools maps those calls into the "look and feel" of a particular GUI. For example, a menu call made by a SQL*Forms application is translated into the appropriate scroll bar of the GUI being used. Oracle applications will therefore have the native "look and feel" of a workstation's GUI.

Oracle's Adaptable User Interface will allow users to keep pace with advances in GUI technology. As new GUIs and advanced features of existing GUIs appear on the market, they can be incorporated into the AUI, allowing Oracle applications to take advantage of them.

Portability Across Data Management Systems

With Oracle, applications may be developed to access data in different data sources with little or no modification. An application that uses SQL to access data in an Oracle database on a UNIX system, can be ported to a VAX/VMS system to access information in RMS files, through SQL as well. Oracle offers portability across data managers by providing a single interface, namely SQL, to different data managers. The technology that enables this is Oracle's data gateway products, called SQL*Connect. The SQL*Connect product accepts the SQL call from the application, and then translates that call into

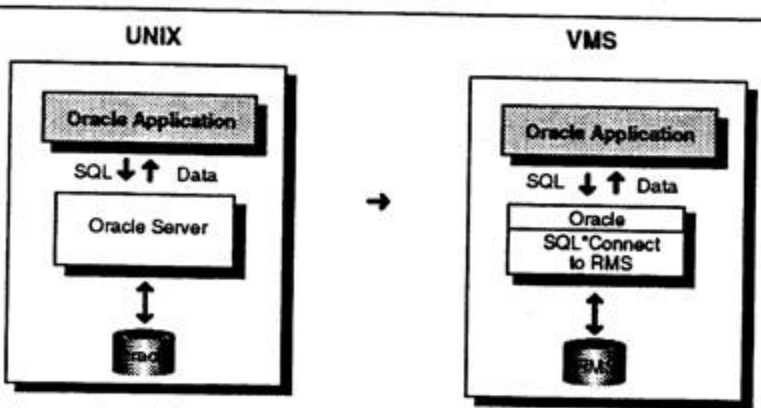

FIGURE 11 *Oracle supports portability across database types.* The same Oracle application can be moved from an Open system to a proprietary system, and be recompiled to run on the proprietary system and access the non-Oracle database through Oracle's SQL*Connect technology.

the appropriate calls of the target data manager. SQL*Connect works in conjunction with an Oracle RDBMS. For example, SQL*Connect to RMS accepts an SQL statement from an Oracle application, it then translates this statement into the semantically equivalent set of RMS input-output file calls, to select the appropriate data set from RMS files.

➡ Oracle Client-Server Interoperability

The ORACLE RDBMS uses a client-server architecture. Oracle's data manager is implemented as a separate process from any client applications supplied by Oracle or built using Oracle tools. This separation of the client application process from the data manager [server] process allows Oracle applications and databases to easily reside on different machines connected on a network.

Oracle's client-server computing solution works over many different types of networks, using different protocols. Oracle's solution also works in multi-protocol environments.

*SQL*Net - Client-Server Over Heterogeneous Networks*

SQL*Net is the Oracle product that enables applications and data management systems to cooperatively process tasks in a networked environment. It allows Oracle client applications to reside on one computer while accessing data from a database on another computer.

Protocol Independence

SQL*Net is implemented in a layered-fashion, so that the Oracle client application or data manager is completely shielded from dealing with any of the details of the network. The advantage of this implementation is that applications can be developed to run in networked environments as easily as if they were being developed on a single system.

Machine Independence

SQL*Net also provides machine independence. Different computers represent data differently. An IBM system represents text data using EBCDIC character codes. UNIX systems use ASCII. The byte orders of data on different computers is different. The floating point formats are different. A problem arises when a client application on a computer that uses one data format accesses information from a data server on a computer that uses a different format. SQL*Net automatically translates data from its source to its destination format. For example, developers of applications needn't be concerned with translating data format differences in a networked environment; this is done automatically by the SQL*Net product making application development much easier.

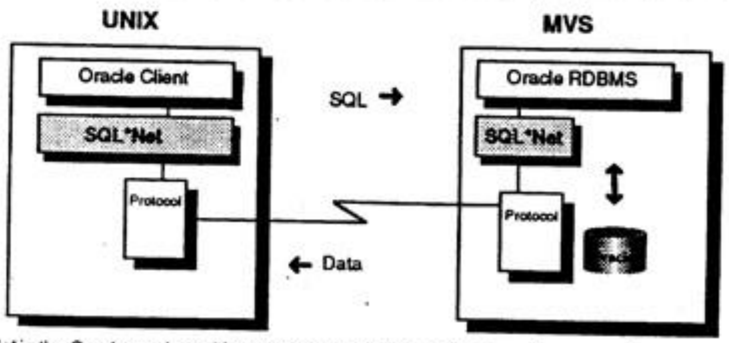

FIGURE 12 *SQL*Net* is the Oracle product which allows Oracle clients and servers to communicate over a network.

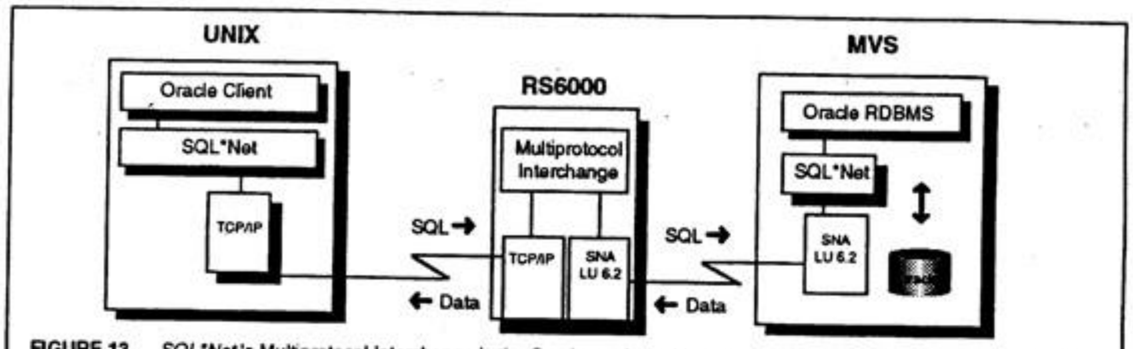

FIGURE 13 *SQL*Net*'s Multiprotocol Interchange is the Oracle product which allows Oracle clients and servers to communicate over two or more different networks.

Running Client- Server Over Multi-protocol Networks

Finally, it may be the case that the client application and the data server are located on networks of different types. For example, a UNIX workstation on a TCP/IP network may need to access data in an Oracle database on an IBM mainframe, where the mainframe is connected to an SNA LU6.2 network. In order for the client to be able to send an SQL message to the data server, the SQL message must be translated from a TCP/IP packet format to an LU6.2 message format [data stream], and conversely, when the requested data is returned. It is SQL*Net V2's multi-protocol interchange capability that performs this translation.

➡ Oracle Open Gateways

Oracle supports a comprehensive Open Gateway strategy. Oracle applications are able to access data in both other relational databases and in non-relational data sources. A strategy is provided to allow customers to build gateways to proprietary file systems and databases, and to access data through a procedural interface. We will now discuss Oracle's Open Gateway strategy.

SQL*Connect - New Relational Applications Accessing Existing Data

A good coexistence strategy allows new SQL-based fourth generation language applications on open systems to be able to access existing data on proprietary systems. Such access is enabled through Oracle's SQL*Connect products. What SQL*Connect does is translate SQL

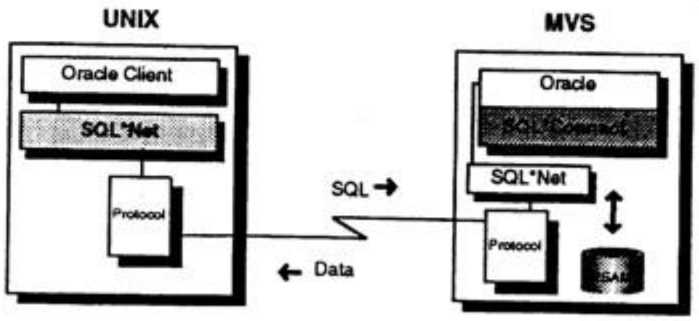

FIGURE 14 *SQL*Connect* is the Oracle product which allows Oracle SQL applications to access data in non-Oracle data sources.

requests into the language of the target data manager. This provides Oracle SQL access to non-Oracle relational databases and even to non-relational data sources.

All Oracle applications, or applications developed using Oracle's fourth generation language tools are designed to work with the SQL*Connect products. SQL*Connect products are implemented as a server process separate from client application processes. Hence, any Oracle client application can work with any Oracle SQL*Connect product. Access to relational and non-relational databases is provided by SQL*Connect products as indicated in Figure 15.

Operating System	Relational	Non-Relational
MVS	DB2, SQL/DS	IMS, VSAM, ADABAS, IDMS
VMS	Rdb, Ingres	RMS
MPE/XL		Turbo Image
ICL VME		IDMSX, ISAM
Tandem	Non-stop SQL	
DG AOS		INFOS II
Wang VS		DMS
Seimons BS2000		SESAM, ADABAS

FIGURE 15 Summary of SQL*Connect Gateways

For availability of each of these Gateways, see your Oracle Sales Representative.

The Oracle Transparent Gateway Developers Kit

Oracle's objective is to provide direct access to 90% of corporate data through its SQL*Connect products. However, some data resides in proprietary files systems for which no SQL*Connect product exists. Oracle will provide its SQL*Con-nect technology in a kit, to allow customers to build their own SQL-based gateway. Whether data is stored in a relational database, in a record-based data manager, or a hierarchical or network database, the SQL*Connect technology can be used to build a data gateway which allows SQL access to the target data manager.

The Oracle Procedural Gateway

A large portion of the world's data is accessed directly by an application procedure. To this end, Oracle will provide the ability to build a remote procedure call (RPC) interface to existing transaction processing systems and applications that have a procedural interface. These include such systems as: TUXEDO, CICS, UTM, DB2 stored plans, IMS-DC transactions, and other user code and programs. In this way, existing transaction processing monitors, like CICS, can be used directly by Oracle fourth generation applications.

Oracle "Embedded SQL Products - Allowing Existing Applications to Access New Relational Data

Oracle provides two strategies by which existing applications can access new Oracle RDBMS. First, existing applications can be partially rewritten, by embedding SQL calls within them, to allow access to

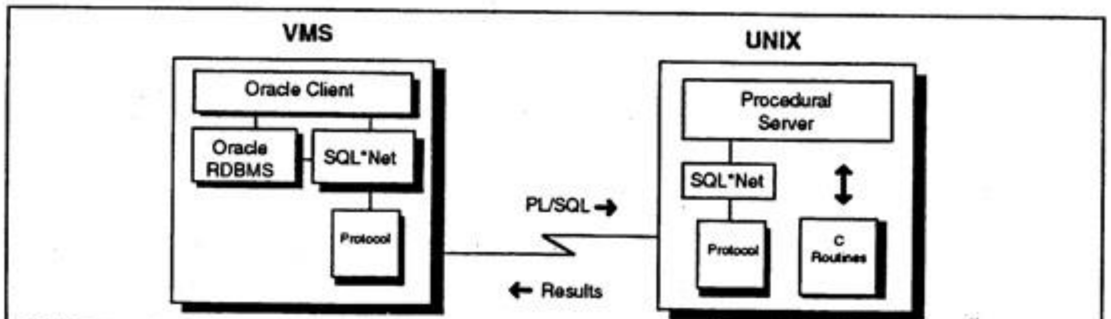

FIGURE 16 *Data available in application servers is available through a procedural interface where a procedure server converts Oracle PL/SQL calls into C routine calls. The procedure server is built through an "open procedure server gateway" kit provided by Oracle.*

relational databases. Secondly, Oracle will be providing Open Gateways which actually translate the calls of existing VSAM and DB2 applications to Oracle SQL calls, so that these applications can access data in Oracle RDBMSs transparently.

Third Generation Pre-compilers

Much application investment has been made in third generation programs on proprietary systems. A typical installation will have several years of COBOL or FORTRAN applications written for their proprietary platform. Applications developers may embed SQL statements within the source of their third generation language programs to access data in relational databases. For example, a organization with a large investment of COBOL applications on an IBM Mainframe could modify those applications to embed SQL statements to access an Oracle relational database on a UNIX system over SQL*Net. Oracle's 'embedded SQL' products are precompilers that compile the source of the embedded SQL statements into third generation language source structures. The source to the entire third generation program is then compiled into an executable per usual.

Oracle has 'embedded SQL' products for the following third generation languages:

- Embedded SQL for COBOL
- Embedded SQL for FORTRAN
- Embedded SQL for IBM PL/1
- Embedded SQL for C
- Embedded SQL for ADA
- Embedded SQL for PASCAL

Transparency Products - Allowing Existing Applications to Access New Relational Data

DB2 Transparency

The second way existing applications may access new, relational data will be through Oracle's 'transparency' products. Planned are 'transparency' gateways for DB2 and for VSAM. The DB2 product will allow DB2 applications on IBM mainframes to access Oracle relational databases, for example, on UNIX. The Gateway will accept DB2 SQL statements from a DB2 client application program and generate the semantically equivalent Oracle SQL statements and pass them to the Oracle server. The Oracle server will process the SQL request, and return the results to the Gateway. The Gateway then will translate the data types from Oracle data types to DB2 data types, and return the information in native format to the requesting application.

VSAM Transparency

A VSAM transparency product is planned as well. This product will allow a native VSAM application to access data in an Oracle database. The product is a Gateway which translates VSAM file input-output calls into the semantically equivalent Oracle SQL statements. This product will allow the great volume of VSAM applications to access Oracle databases on MVS, or UNIX, or other platforms, with no modification.

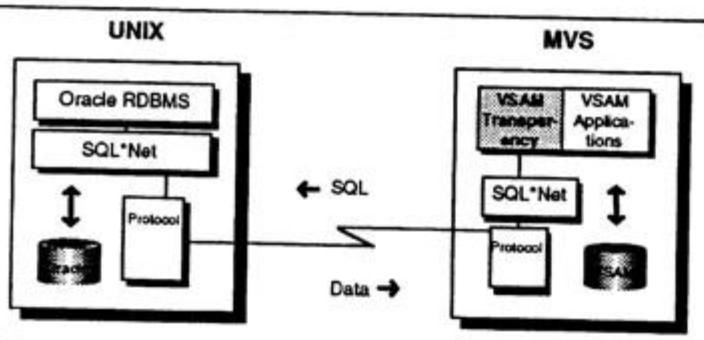

FIGURE 17 *VSAM Transparency* products translate VSAM input-output calls into SQL statements that are processed by an Oracle RDBMS

➡ ORACLE Cooperative Server Database

Finally, it was discussed how the 'Cooperative Server Database' approach could be used to allow multiple systems networked together to act as a unified data processing system for user queries and transactions.

ORACLE7 allows Cooperative Server Database to be created to process data in multiple Oracle and non-Oracle databases.

Oracle Homogeneous Cooperative Server Database

There are two basic operations performed by Oracle's Cooperative Server Database capability: distributed queries [collecting data from multiple sites], and distributed updates [updating data at multiple sites].

Distributed Queries

Oracle allows data in databases on different machines to be efficiently collected and correlated by users through its distributed query capability. This distributed query capability allows multiple machines, for example, UNIX systems and mainframes, to work together to collect data from multiple databases for users, creating a unified data processing environment. In ORACLE7, Oracle queries execute in the quickest possible way due to a distributed query optimizer.

Distributed Updates

ORACLE7 supports distributed updates. Distributed updates allow user applications to execute transactions that update data stored on multiple machines. Users can treat data on multiple machines as if it resided on a single system, providing a unified computing environment.

ORACLE 7 RDBMS supports a facility called 'automatic two phase commit,' which automatically insures the integrity of a transaction in a computer environment even when machines, networks, or database servers fail. It works by verifying whether all tables referenced in a transaction are available to be updated, before it proceeds with the update. This two phase verification process is necessary to insure data integrity should a system, database, or network crash during the transaction. Additional facilities are provided to automatically recover multiple databases into a consistent state, after a crash has occurred.

Oracle Heterogeneous Cooperative Server Database

The second type of Cooperative Server Database discussed was a 'heterogeneous Cooperative Server Database;' one where data is collected from both Oracle and non-Oracle databases. The ORACLE7 RDBMS is able to make data distributed in both Oracle and non-Oracle databases appear as if it were all stored in a single database. It does so by the fact that the Oracle RDBMS is designed to work with the

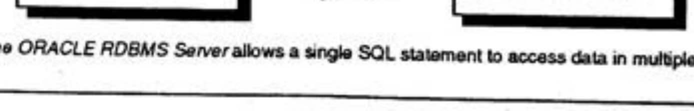

FIGURE 18 *The ORACLE RDBMS Server* allows a single SQL statement to access data in multiple Oracle RDBMSes.

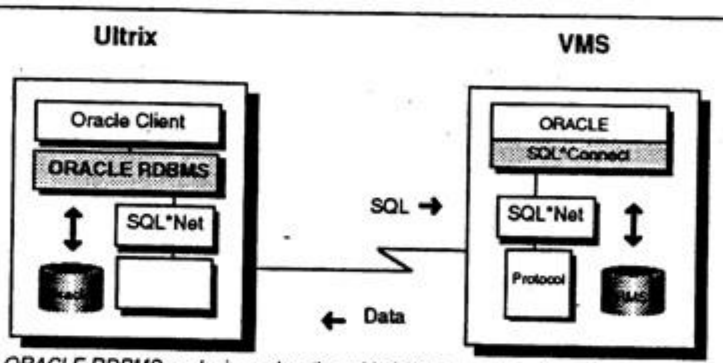

FIGURE 19 *The ORACLE RDBMS works in conjunction with SQL*Connect to allow Oracle and non-Oracle databases to be accessed by a single SQL query, or updated (V7) by a single transaction.*

SQL*Connect gateways to provide `heterogeneous Cooperative Server' data access. For example, a user on a UNIX system may issue a query which joins data in a table in an ORACLE relational database on the UNIX system, and an RMS file on the VAX.

ORACLE7 Heterogeneous Distributed Updates

ORACLE7 supports distributed updates against heterogeneous databases as well. This means that users can execute transactions that update data in existing databases on proprietary systems in conjunction with Oracle databases on open systems platforms. Existing and new systems cooperate as a single transaction processing system.

In addition, Oracle offers some advanced coexistence features provided by ORACLE7 which extend Oracle's Cooperative Server Database features even further

Advanced ORACLE7 Coexistence Features

ORACLE7 RDBMS offers a set of new features that allow a much tighter coexistence strategy. Among these are 'snapshots,' `replicate copies,' `remote procedures, functions, and packages,' `remote triggers,' and `remotely triggered procedures.'

➠ Snapshots

A 'snapshot' is a copy of data from one system to another. The copied data may be a selected subset of data in one or more tables. These copies of data can be done at different times. Snapshots are useful to automate the dissemination of information. What was done manually with SQL*Loader now can be done automatically with snapshots. As an example, employee information may need to be periodically downloaded from the corporate employee database to the employee databases stored at each of the individual factory outlets. In this case a snapshot of personnel data about employees may be selected from each region from the corporate database on the mainframe and automatically downloaded into the databases on the remote systems.

➡ Remote Procedures

The ORACLE7 RDBMS allows the creation and storage of procedures within the database. A procedure is a set of statements to be executed according to some procedural logic. Procedures stored in remote databases may also be called and executed. For example an application on one platform, say UNIX, may call a procedure which is stored in a DBMS on another platform, say VMS. Not only do procedures provide cross-operating system procedural access, but procedures allow Oracle applications to access other DBMSs with procedural interfaces, or even non-DBMS applications, with procedural interfaces.

➡ Remote Triggers

In Oracle, a 'trigger' is a defined action that takes place when a named field is changed [updated, deleted, or inserted]. That action may be a SQL statement or a series of SQL statements executed according to some procedural logic, i.e., a procedure. In ORACLE7, defined triggers are stored in the database, as are the procedures which they may call. When the action triggered is remote, the trigger is a 'remote trigger.' Remote triggers may be used to perform cross system value-based auditing, enforce complex business rules that involve databases on open and proprietary platforms, implement complex security rules, and automatically make implied changes.

➡ Replicate Copies

'Replicate copies' are entire copies of a table at one site which is copied to one or more other sites. These copies of data are 'synchronous,' i.e., all data is copied at the same time. With Oracle, replicate copies are done by creating a 'trigger' which reproduces changes to the master copy in each replicated table. The multiple updates to the slave tables are performed as a distributed transaction, so that they are protected by two-phase commit should there be a failure during the transaction. So, for example, if a user at the mainframe site updates the master price table, this price change is simultaneously propagated to the price tables of the databases on UNIX systems in the retail outlets. This is done by defining a trigger on the price field of the table in the mainframe database, such that whenever that field is updated, that update is propagated to the same field of the same table in the UNIX databases at each of the retail outlets. In this way, all 'copies' of the mainframe database table at the UNIX sites are kept up to date.

Figure 20 summarizes the Oracle products which implement the various coexistence approaches discussed.

Approach		**ORACLE Product**
Upload/Download	Manual Data Transfer	SQL*Loader
Application Portability	Across operating systems Graphical user interfaces Data management systems	All Products Adaptable User Interface SQL*Connect
Client-Server	Across heterogeneous networks Across an internetwork	SQL*Net MultiProtocol Interchange
Open Gateways	New applications access existing data Access to procedural servers Existing applications access relational data	SQL*Connect Open Procedure Gateway Embedded SQL, Transparency Product
Cooperative Server Database	Homogeneous Heterogeneous	ORACLE & SQL*Net ORACLE, SQL*Net, & SQL*Connect

FIGURE 20

Summary

In summary, the several distinct advantages of open systems make it attractive for organizations to purchase them, and incorporate these into the existing mix of machines. At the same time, it is important for these organizations not to lose their investment of data, applications, and training in the proprietary minicomputers and mainframes they already have installed. The key is to forge a 'coexistence strategy' which allows the proprietary minis and mainframes to continue to be used in conjunction with the newer Open systems.

It is important to understand that a coexistence strategy is not an all or nothing proposition. Coexistence may be implemented in different ways and to different degrees. A good coexistence solution offers flexibility as to how exactly coexistence is implemented. It may be that merely periodically transferring data between the UNIX and proprietary systems is an adequate solution. It may be that offloading application processing from the mini or mainframe is what is required. Here, the UNIX systems may be used for application processing, while the mainframe retains the data. A client-server solution may be implemented to allow the UNIX applications to have on-line access to existing data. Or, more sophisticated solutions may be implemented that create a 'Cooperative Server Database' between the proprietary and open systems.

Regardless of the coexistence path chosen, the overall goal remains the same: continue to use existing proprietary systems in concert with new open systems technology. To this end, Oracle provides a comprehensive product set to implement flexible coexistence solutions.