

APPENDIX B TECHNOLOGY ABSTRACTS

This appendix provides further detail regarding technologies used within the candidate framework architectures. These overviews are not exhaustive; rather they intended to provide a management overview only. Specific technologies or technology classes described within this appendix are:

- Middleware.
- X Window System.
- Transport Control Protocol/Internet Protocol (TCP/IP).
- Executive Information Systems/Decision Support Systems (EIS/DSS).
- Relational Database Management Systems (RDBMS).
- Data Warehousing.
- Operating Systems.
- Electronic Data Interchange (EDI).
- Network Infrastructure Technologies.
- Online Transaction Processing (OLTP) Monitors.
- Web-based Distributed Systems.

TECHNOLOGY: Middleware

TECHNOLOGY DESCRIPTION:

Middleware¹ provides distributed application component connectivity through a set of operating system and network services. These services are typically made available to the developer via an application-programming interface (API)². Using middleware, developers need not be masters of multiple communication technologies – an undertaking that would require considerable engineering effort. Instead, middleware isolates application developers from the details of complex communication protocols, services, and interfaces. Developers are thus able to focus on solving business problems. Many of the features and facilities provided by middleware have existed for many years. However, the integration of these services under a unified construct that operates without regard for operating environment and network infrastructure distinguishes middleware and gives it value.

Middleware can be broadly categorized as either vendor-specialized or generalized. Each of these software classes are appropriate within certain problem domains, but only standards based generalized middleware can enable truly open solutions.

Vendor-specialized middleware is inherently proprietary and typically used within remote data access client/server architectures -- commonly referred to as Remote Data Access (fat client) solutions.

Middleware within this classification is generally provided as an extension of a vendor's server product, such as a relational database management system, and **only** works with that specific product³. Although these products do isolate the application developer from the complications of distributed interprocess communication, their use does not result in an open systems architecture.

On the contrary, these middleware

solutions merely trade the hardware dependencies found in the monolithic systems for a new reliance on vendor-specific software solutions -- the result is **software lock-in**⁴. This is not to say vendor-specialized middleware products aren't viable solutions. These technologies tend to be easier to use than generalized middleware technologies and don't further compromise "openness" in those architectures where the

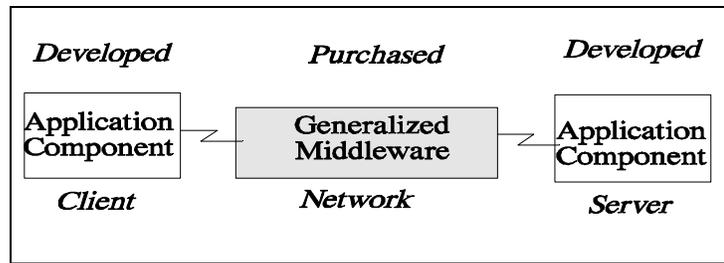


Figure B-1. Three Tiered Client/Server Architecture

¹The terms "glueware" and "socketware" are also used to refer to the "middleware" software category.

² These services generally provide functionality defined within the top three layers of the Open System Interconnection reference model (OSI). The OSI is a framework for defining standards that facilitate interconnection of heterogeneous, open operating environments. Specifically, the model is concerned with the capability of systems to cooperate by being interconnected through some standardized communications facility and by executing standardized protocols. The top three OSI layers define application, presentation, and session interconnection services.

³Examples of vendor specialized middleware include Oracle's SQL*Net and Sybase's Open Server/Open Client products.

⁴It should be noted that this argument is made by many within the IT industry regarding the use of stored procedures and SQL that is not ANSI compliant. As a result of these complications, practical tradeoffs must currently be made between performance and openness.

proprietary features of server products are being used (often a necessity if acceptable performance is to be achieved).

Generalized middleware tends to be based on symmetric communication mechanisms that are independent of vendor-provided server products. As a result, these technologies offer enhanced application design flexibility and can be used to develop vendor-independent functions that support a variety of client/server distribution strategies. This flexibility is made possible through a set of services that enable application components to:

- Interoperate without regard to the underlying computer architecture. This is made possible by isolating application components from operating environments and associated infrastructure. This isolation is facilitated via an abstract interface, the middleware API, which decouples application components from native operating environment implementations, resulting in more portable applications.
- Interoperate without regard for the underlying network architecture. This capability, often referred to as "context bridging" or "transport bridging", facilitates transparent communication between operating environments using heterogeneous network protocols. This capability is necessary if application components are to be distributed across disparate networks, for example, TCP/IP, SNA, IPX/SPX, and X.25.
- Share data and function parameters without regard for the underlying operating environment, and networks. Typically, solutions providing this capability must provide "data translation" and "parameter marshaling" services. Data translation services insure that data is presented to distributed application components in their native format. For example, data may need to be translated from EBCDIC to ASCII, between different byte orders, or between data type representations. Parameter marshaling involves preparing data structures for network transmission and use by distributed application components. For example, when working with pointers, parameter marshaling services are responsible for identifying the referenced data, copying the data and preparing it for network transmission, unpacking the data after transmission, and passing it by value to the distributed application component.

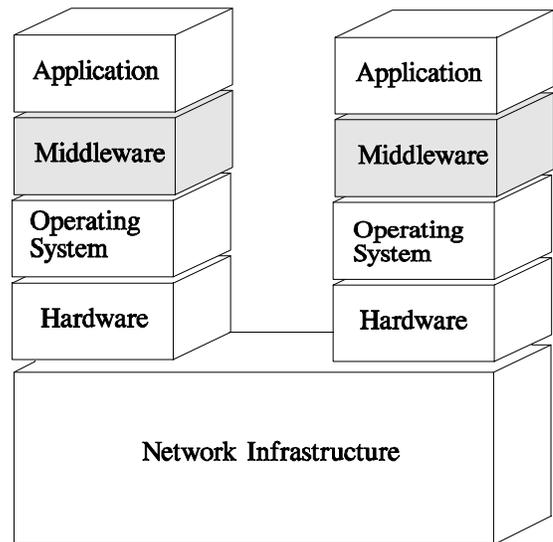


Figure B-2. Generic Network Protocol

Most commercially available middleware solutions are based on the remote procedure call or messaging/message queuing technologies. In terms of service, all of these models provide essentially the same function: they facilitate the interprocess communication of data between distributed application components. However, there are important functional differences, varying levels of standardization, and degrees of complication associated with each of these technologies.

The remote procedure call (RPC) is the most mature of the general purpose middleware technologies and uses the familiar concept of procedure calls to facilitate interprocess communication between distributed application components. RPC middleware products typically provide "high level" communication services, such as data translation and marshaling, and can be characterized as inherently synchronous and blocking. Consequently, RPC-based middleware usually requires that the:

- Client and server be simultaneously available at the time of service request,
- Client wait until the server replies before proceeding to other tasks, and

- Developer use complicated API suites -- including as many as 600 verbs.

RPC-based middleware is functionally rich. However, because RPC communication is generally synchronous and blocking, it is not always an appropriate interprocess communication solution. One alternative to RPC technologies is messaging-based middleware.

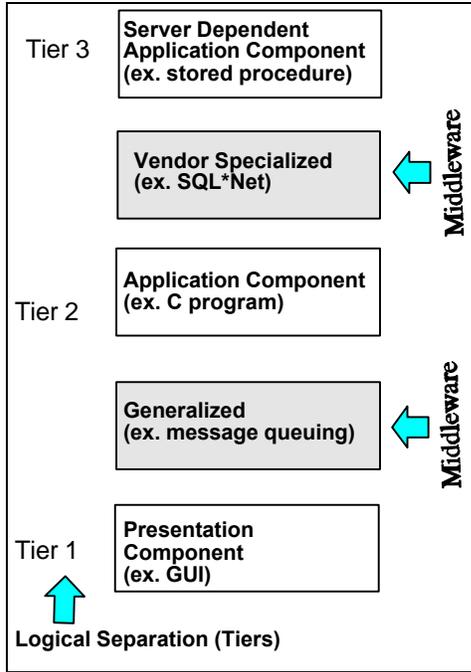


Figure B-3. Messaging-Based Middleware

Messaging/message queuing technologies use message passing and queuing to enable application interprocess communication. The message-passing model allows the client to call an API and send a request in the form of a message to the server. In many implementations, the message-passing model is enhanced via queues that facilitate extremely flexible solutions and truly asynchronous communication. This asynchronous model, coupled with comprehensive transport bridging services, makes messaging ideal for use with wide area network technologies -- where geographically distributed resources may not be simultaneously available. However, like RPC, messaging middleware is not "the" distributed communication solution. In fact, messaging middleware has several shortcomings -- one of which is lack of standardization. That is, messaging technologies are at best, void of industry and defacto standardization; and at worst flat out proprietary⁵. Additionally, messaging technologies provide only "lower-level" communication services. Consequently, the developer is burdened with creating data translation and parameter marshaling services.

No single middleware product does it all. Within large systems, it is likely that more than one middleware technology will be required to support interprocess communication and affect required performance, security, fault tolerance, and application flexibility. For example, the use of both vendor- specialized and generalized middleware solutions is commonplace within distributed system architectures. Likewise, the use of both RPC and message queuing middleware technologies is often required to adequately support business solutions.

COMMERCIAL OFFERINGS:

Vendor	Product
IBM	MQ Series, Component Broker Connector (CBConnector), Component Broker Toolkit (CBToolkit)
Momentum Software Corporation PeerLogic	XIPC, MessageExpress, Visual Flow+ Named Pipes

⁵ The Message Oriented Middleware Association (MOMA) is currently working to develop messaging middleware standards.

TECHNOLOGY: X Window System

TECHNOLOGY DESCRIPTION:

For many years, ASCII dumb terminals were used exclusively to access host-based (UNIX, etc.) applications and systems. However, these terminals were incapable of meeting the ever-increasing user demand for graphical output and graphical user interfaces. In an effort to satisfy these demands the Massachusetts Institute of Technology developed the X Window System (X11). The X Window System is a network-based graphics engine that allows users to connect to, and execute applications on, remote systems while handling/managing local I/O (mouse, keyboard, display etc.) interaction.

X11 allows the user to execute and display many applications simultaneously, each within one or more windows. The display is controlled by software that is referred to as the X “server.” Applications, which are referred to as X “clients,” do not interact with the user’s display directly. Rather, X clients transmit requests to the X server, which manages the display on behalf of the X client. That is, unlike the traditional client/server paradigm, with X11, the client process executes on the application “back-end,” handles all application data processing except I/O interaction, and is a shared resource that typically runs on a server class machine. Whereas, the X display server manages I/O and user-interface services, typically executes on a client platform, comprises the application “front-end,” and controls the user’s display (including the screen, keyboard, and mouse). These components are illustrated in Figure B-4 and are further described in the following paragraphs.

The X server controls the display hardware. It, or the underlying operating system, contains device drivers for the keyboard, mouse, and screen. As such, the “server” contains the only hardware dependencies within the X11 system. The server accepts *requests*, which are sent across the network from applications - the X client(s). These requests can include – among other things – instructions to create windows on the screen, to change the size and location of windows, or to draw text or graphics within displayed windows. In addition to servicing client requests, the server sends *events* to the client. These events provide the client with keyboard or mouse input and provide window status information. As Figure B-5 illustrates, a server can manage multiple, simultaneous client connections.

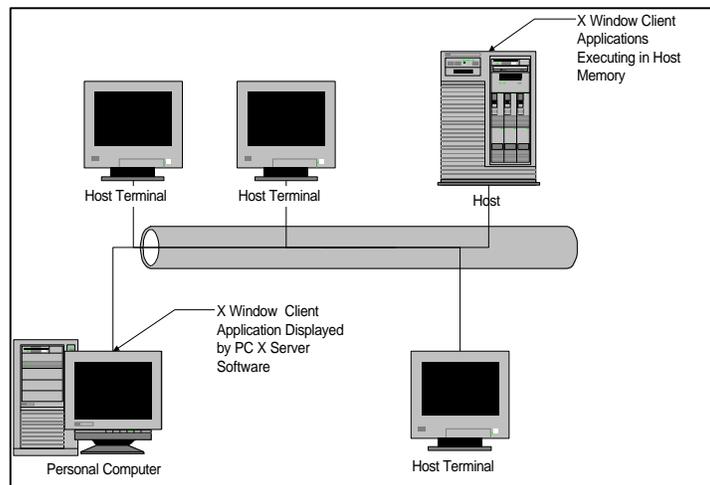


Figure B-4. Components of X Window System

When originally introduced, X servers and X terminals were synonymous. That is, X server services were supplied exclusively via terminals manufactured to handle X11. However, today, X server services can be delivered to users on a variety of platforms. Among the relative newcomers to the X11 market are PC-based X server implementations. PC X Server software provides X server services to personal computer users. Simply put, PC X servers provide concurrent access to multiple host-based applications (X clients) from within Windows, Windows95, Windows NT, and other PC environments; while simultaneously allowing the user to access PC-based applications, such as office automation and productivity software.

Unlike the X server, the X client does not directly manage the users’ display. The client program contains the software required to perform the function the user requires – process an application, send electronic mail, maintain a database, etc. Additionally, the client includes the software required to support X11 and to

provide the graphical user interface. In the parlance of this document, the X client provides application and data management logic, as well as some portion of the application's presentation logic. A client can operate with any display (without recompiling or relinking the application), as long as there is a suitable communications link and an X11 compliant server available to manage the user's display.

As illustrated in Figure B-5, the communications link is the third major component of the X Window System. This communication link can be a local or wide area network – X will run across almost any type of network: TCP/IP, DECnet, IPX/SPX, Ethernet, Token Ring, X.25, serial lines, etc. In addition to executing remotely, X clients can also be execute locally – on the same machine as the X server. In cases where the X client and server are executing on the same machine, the communication link can be any available form of inter-process communication, including shared memory, named pipes, UNIX sockets, etc.

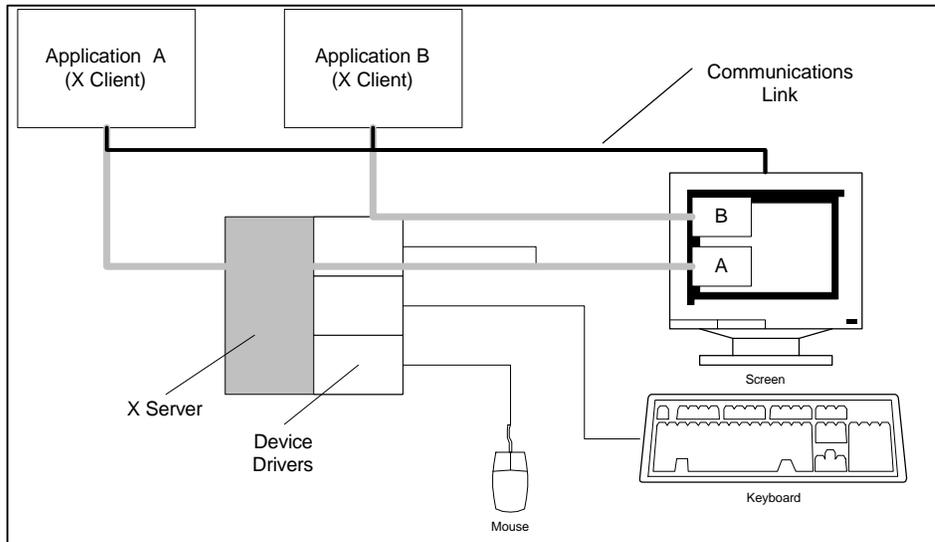


Figure B-5. Communication link of X Windows System

The ability to access both locally and remotely executing X applications is referred to as *network transparency* – that is, regardless of where the X client is executing, X11 creates the illusion that the application is running at the user's machine. This is particularly useful in situations where an application must run on a particular machine, but it isn't feasible to give all users a machine of this type. For example, an organization's management information systems may require an enterprise class server or mainframe. By using X11, all users can execute the application on the server, but interact with it and display output on substantially less powerful and less expensive workstations. (Note: This physical system distribution strategy is described as "remote presentation" within Section 2 of this document.)

Originally, X11 was implemented for the UNIX environment; however, X11 is now available on a variety of platforms, ranging from mainframes to personal computers. For example, the following is a partial list of operating environments that currently support the X Window System and X-enabled applications:

- | | |
|--|------------------------------------|
| • AT&T UNIX System V | • NeXT |
| • Apple Macintosh MacOS and A/UX | • VAX VMS |
| • Cray UNICOS | • SunOS, Solaris on Ultras, SPARCs |
| • Hewlett Packard HP-UX on HP9000/s300 | • X terminals from various vendors |
| • IBM AIX on PS/2 and RS/6000 | • SCO UNIX |
| • IBM Mainframes, MVS | • NeXT |
| • Microsoft Windows, Windows95, and Windows NT | • Data General DG/UX |

Figure B-6. Operating Systems Which Support X-Enabled Applications

X11's network transparency and openness allows users to access applications running on heterogeneous operating environments. This provides users with a rich and flexible work environment, as all applications can be delivered to users regardless of the X client or X server operating environment heterogeneity. This is illustrated in the following figure.

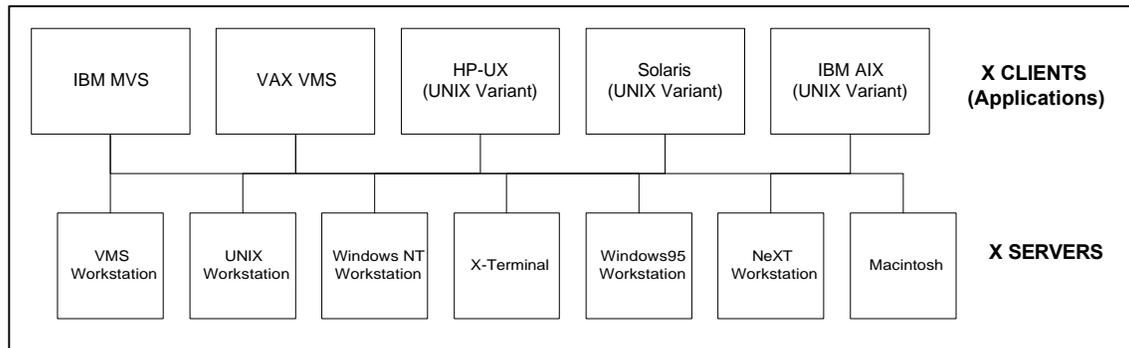


Figure B-7. X Clients and X Servers operating environment

Another advantage the X Window System offers is improved manageability. The X Window System allows the Remote Presentation system distribution strategy to be used. With this strategy, only the X server software is located and executing on the client platform. This simplifies system modification, configuration management, and software distribution, as application software does not reside on the users machine (the client). This configuration also helps to minimize client platform operating resource requirements, as only a minor portion of the application (the X server) executes on the client platform.

COMMERCIAL OFFERINGS:

Vendor	Product
Hummingbird	EXceed
WRQ	ReflectionX/Reflection 2
SCO	Xvision Eclipse
NCD	PC-Xware

TECHNOLOGY: TCP/IP

TECHNOLOGY DESCRIPTION:

The TCP/IP protocol suite was developed more than 25 years ago, when it formed the underpinning of the government network that has evolved into the Internet. TCP/IP combines a variety of protocols, each of which are responsible for different layers within the 7-layer Open System Interconnection (OSI) model. That is, TCP/IP provides or relies on services that are consistent with 4 layers within the OSI framework. For this reason, it is convenient to describe TCP/IP in terms of the OSI model.

The OSI model is a framework for defining standards that can be used to “link” heterogeneous systems. Within the context of the OSI, “open” denotes the ability of any two or more OSI compliant systems to communicate. OSI is intended to facilitate communication and information exchange through any standardized communication facility executing standardized OSI protocols.

The OSI model is composed of communication functions, which are partitioned into a hierarchical set of layers. Each of these layers performs a subset of the functions required for system communication. Within this hierarchical construct each layer relies on the next layer to perform more primitive functions and to conceal the complexity of those functions. The OSI model is illustrated in Figure B-8 and is defined in the following paragraphs.

- **Application Layer.** Provides services to the users of the OSI environment; TCP/IP examples include file transfer protocol (FTP), remote login (telnet), Simple Mail Transfer Protocol (SMTP), and Simple Network Management Protocol (SNMP).
- **Presentation Layer.** Performs data transformations required to provide a standardized application interface and provide common communication services; Examples include encryption, External Data Representation (XDR), and the X Windows protocol; TCP/IP does not provide Presentation services.
- **Session Layer.** Provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between cooperating applications; Hyper-Text Transport Protocol (HTTP) provides Session layer services; TCP/IP does not provide Session services.
- **Transport Layer.** Provides reliable, transparent transfer of data between end-points; Provides end-to-end error recovery and flow control; Ensures that data is delivered error-free, in sequence, with no losses or duplications. The Transport Control Protocol (TCP) provides transport layer services. That is, TCP is responsible for verifying the correct delivery of data from client to server. Additionally, because data can be lost or corrupted during transmission, TCP provides error detection and retransmission features. That is, TCP acknowledges the reception of packet, sequences packets for

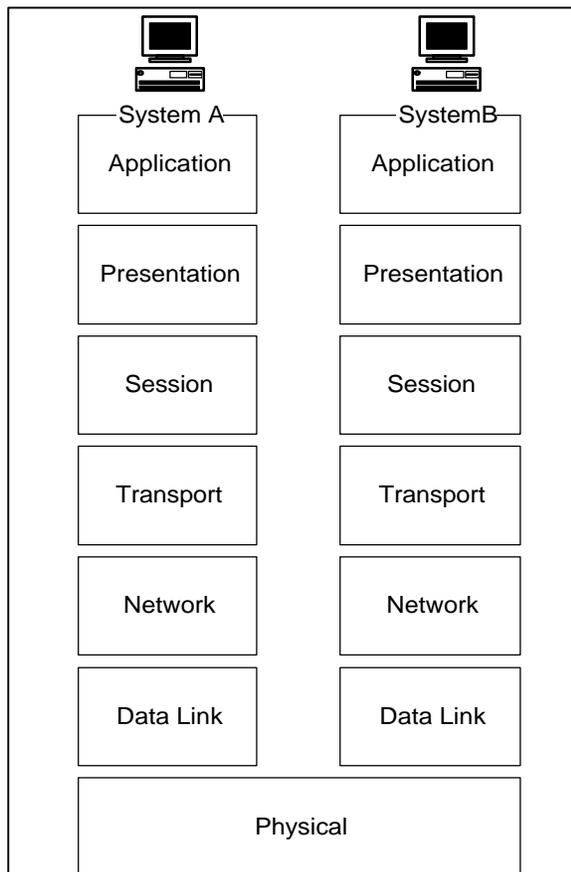


Figure B-8. 7 layer OSI model

further processing, and request retransmission of missing or damaged packets. These features ensure that transmitted data packets are completely and correctly received, and that only missing or damaged packet are retransmitted. These features allow TCP to provide reliable and efficient transport services, even through the service it uses (IP) is unreliable.

- **Network Layer.** Provides upper layers with independence from the data transmission and switching technologies used to connect systems. The Network layer is responsible for establishing, maintaining, and terminating connections. Additionally, the network protocol facilitates the switching and routing functions necessary to allow distributed system communication. The Internet Protocol (IP) provides Network layer services and is responsible for preparing data for network transmission and moving packets of data between network nodes. However, IP does not specifically determine the network route to be traversed by data packets, nor does IP ensure packet delivery. Rather, IP relies on other communication services – namely TCP – to ensure that data is delivered and that delivered data is complete and error free.
- **Data Link Layer.** Provides for the reliable transfer of data across the physical network link; sends blocks of data (frames) with the necessary synchronization, error and flow control. Examples include Ethernet, and Token Ring. Specifically, these protocols organize data into a set of frames and supplement each frame with control bits that allow for reliable data transmission. TCP/IP does not provide Data Link services, but does rely on these services.
- **Physical Layer.** Supports the transmission of unstructured bit streams over physical network links; deals with the mechanical, electrical, and procedural characteristics used to establish, maintain, and deactivate the physical network connection. Examples include unshielded twisted pair, broadband coaxial cable, and baseband coaxial cable. TCP/IP does not provide physical level services, but does rely on these services.

As previously mentioned, the Internet Protocol (IP) was developed to create a “Network of Networks” – the Internet. To facilitate this interconnection, TCP/IP was designed to leverage the many disparate network technologies used to interconnect systems. For example, TCP/IP can be used on large internal networks based on IBM’s System Network Architecture, or with relatively new technologies, such as Integrated Service Digital Networks (ISDN), frame relay, Fiber Distributed Data Interchange (FDDI), and Asynchronous Transfer Mode (ATM).

Each Network layer protocol has its own convention for transmitting messages between two machines within a network. In an SNA network, every machine has Logical Units with their own network address. Similarly, DECnet, Appletalk, and Novell IPX all have facilities for assigning numbers to each local network and to each workstation on the network. TCP/IP assigns a unique number (IP address) to every workstation attached to the network. This “IP address” is used to route packets to the correct network and machine. IP addresses are 4 byte values (32 bits), which are typically expresses via dotted-decimal notation. This notation is determined by converting each byte (within the IP address) into a decimal number (0 to 255) and separating the bytes with a period (for example, 201.81.155.170).

It is important to note that every TCP/IP network node must have a unique Internet address (IP address). For those networks exposed to the Internet, IP addresses are controlled and allocated by a centralized authority -- the Internet Network Information Center (InterNIC).

It’s difficult to accurately gauge what proportion of businesses use a particular protocol. However, according to *Information Week*, when considering the “Big three” networking protocols – TCP/IP, Novell’s Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX), and IBM’s System Network Architecture (SNA) – TCP/IP is becoming more dominant in an increasing number of industries, including the government sector. As illustrated in Figure B-9, TCP/IP acceptance among business users is largely result of TCP/IP’s open architecture and ability to support local area and wide area networks.

For example, TCP/IP is preferred to IBM's proprietary SNA by many organizations, despite SNA's superior security and administration features. This is largely because SNA depends on a mainframe-centric computing environment and requires specialized and often expensive equipment. Whereas, TCP/IP is supported by almost every software and hardware vendor -- for instance, the TCP/IP networking base is the same on all variants of UNIX (AIX, HP-UX, SCO, Solaris, etc.). What is more, TCP/IP administration is becoming easier, thanks to ongoing development of the protocol. For example, technologies like Dynamic Host Configuration Protocol (DHP), which is supported by vendors such as Sun Microsystems and Microsoft, lets users automatically configure and administer client machines from a central server. This reduces administration and maintenance efforts associated with assigning individual TCP/IP addresses to every PC. For these reasons, and others -- including the proliferation of the Internet -- TCP/IP will continue to gain prominence within network-based information systems.

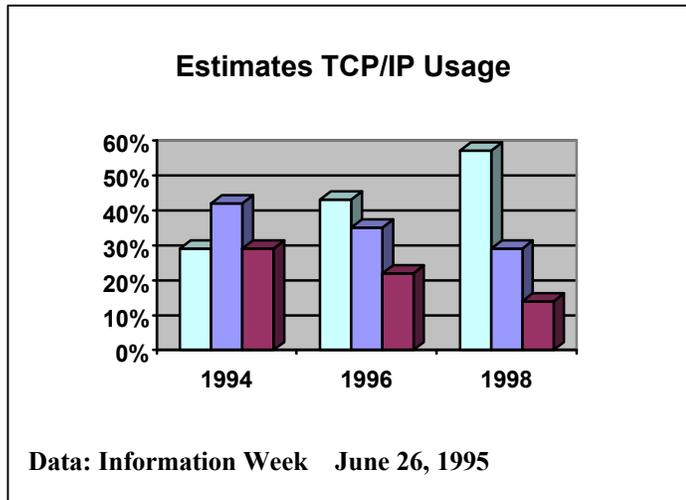


Figure B-9. Estimated TCP/IP usage

COMMERCIAL OFFERINGS:

Vendor	Product
Microsoft	TCP/IP is a native operating service within Windows for Workgroups 3.11, Windows95 and Windows NT
UNIX Vendors (Sun, HP, IBM, SCO, etc.)	TCP/IP is a native operating system service under Solaris, HP-UX, AIX, and all other UNIX variants
NetManage Inc.	Chameleon TCP/IP
Wollongong Group	WIN/TCP for DOS
Novell, Inc	LAN Workplace for DOS
Ungermann-Bass Inc.	Net/ONE TCP BNS/PC
FTP Software	PC/TCP

TECHNOLOGY: Executive Information Systems/Decision Support Systems

TECHNOLOGY DESCRIPTION:

Organizations generate and use tremendous amounts of data and manage this data from a wide range of locations with disparate technologies. Faced with the challenge of effectively and efficiently using this resource, organizations are turning to technologies that facilitate practical consolidation and analysis of this data. These technologies are clearly becoming a significant and important organizational resource and are being used to develop executive information systems/decision support systems (EIS/DSS).

EIS and DSS technologies are typically used by executives and decision-makers to improve decision-making, planning, communications, personal efficiency, and organizational control. These technologies can be characterized as systems that:

1. Are used directly by decision makers and leaders without the assistance of intermediaries.
2. Provide easy on-line access to current information about the plans of the organization.
3. Are designed with management's critical success factors (CSF) in mind.
4. Use state-of-the art-graphics, communications, and data storage and retrieval methods.

Specifically, EIS and DSS technologies provide:

- Easy to use and maintainable graphical user interface requiring minimal or no training to use.
- Integrated capabilities for electronic communications and data access, security and control.
- On request “drill down” capability to lower levels of detail and data analysis.
- Depiction of organizational health indicators using graphical, tabular, and/or textual information.
- Functionality for decision support, ad hoc queries and “what-if” analysis.
- Data analysis, on-line status, trend analysis, statistical analysis, exception reporting.
- Extraction, filter, aggregation, and tracking of critical data.
- Access and integration of a broad range of internal and external data sources, including data warehouses, data marts, and databases.

Obviously these are useful systems with the potential to drastically improve the effectiveness of organizational leadership. However, delivery of these systems can be challenging for system integrators. These challenges can be attributed to difficulties associated with:

- Identifying and accessing information considered most useful to organizational leaders.
- Modeling system requirements, which are derived from executive tasks that are often judgement based, highly unstructured and difficult to describe.
- Understanding the user environment, organizational goals, and CSFs.
- Exploring ways to measure CSFs and devising ways to deliver key information to support these factors.

Simply put, delivery of useful executive information systems requires more than just systems engineering skills. Delivery of these systems also often requires an understanding of statistical and quantitative sciences, management science, and business process reengineering.

EIS/DSS tools can be classified as Query/Analysis, Online Analytical Processing (OLAP), and Data Mining technologies.

Query/Analysis tools allow decision-makers to formulate queries without writing a program or using a structured query language. That is, the tools provide the capability to generate queries through a “point-and-click interface. Additionally, these tools typically provide graphical output -- including bar charts, pie charts, histograms, and line graphs.

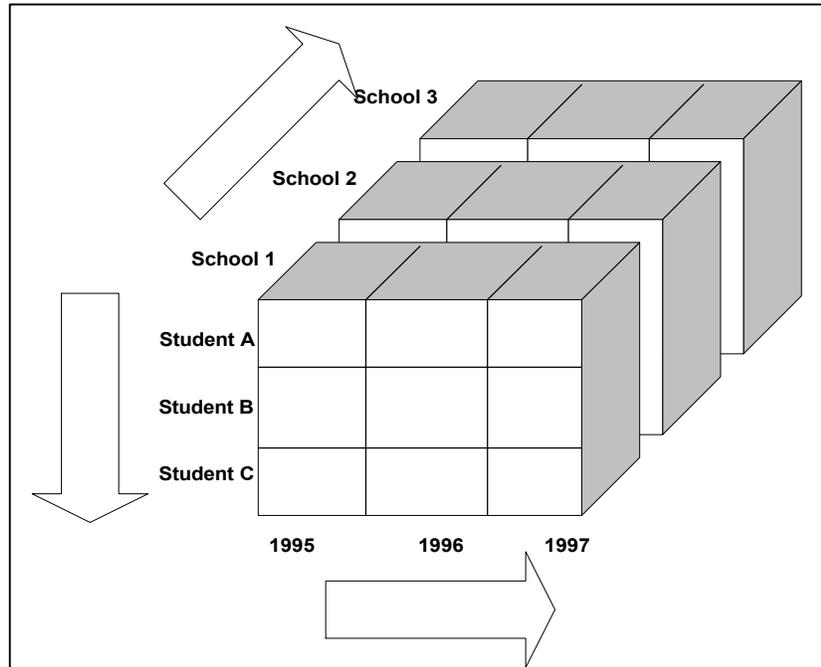


Figure B-10. Multi-dimensional data view

Online Analytical Processing (OLAP) tools create, organize, and format multi-dimensional data views. Typically these views are based on data that is maintained within SQL-based relational database management systems (RDBMS), specialized OLAP SQL databases, or, more recently, multi-dimensional DBMSs. For example, using OLAP technologies, the Education Department could review data via multiple dimensions such as time, student, school, and disbursement. This would allow ED to ask the following question: What are the disbursements – by year, by student, by school? This is illustrated in Figure B-10.

Data Mining tools are the least mature of the EIS/DSS technologies. These tools allow users to search for data patterns and groups within expansive data sets. Unlike the Query/Analysis and OLAP technologies, Data Mining tools do not respond to low-level queries formulated by the decision-maker. Rather, Data Mining tools use search methods, such as data associations and sequence patterns, to discover and present information. Typically, Data Mining output is presented in the form of *if-then* rules, as illustrated in Figure B-11.

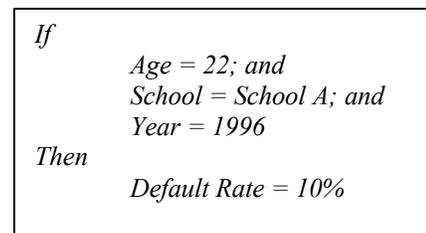


Figure B-11. Output of a Data Mining *if-then* Rule

COMMERCIAL OFFERINGS:

Vendor	PRODUCT	PRODUCT TYPE
Gupta, Inc	Quest	Query/Analysis
SAS Institute	SAS System, SAS OLAP ++	Query/Analysis
IBM Corp.	Visualizer, DataDiscovery	Query/Analysis, Data Mining
Cognos Inc.	Impromptu, PowerPlay	Query/Analysis, OLAP
Oracle/IRI	Express	OLAP
Software AG Inc.	NetMap	Data Mining
Red Brick Inc.	Red Brick Warehouse	OLAP

TECHNOLOGY: Relational Database Management Systems

TECHNOLOGY DESCRIPTION:

A database management system (DBMS) consists of a collection of interrelated data and the programs required to access and manage this data. The primary goal of a DBMS is to provide an environment that facilitates convenient and efficient information usage. That is, database systems provide users with an abstract view of the data, which hides complexities associated with the data storage, retrieval, modification, and other mechanisms.

Relational DBMSs (RDBMS) make use of data structures that are based on the relational model. The relational model describes data and data relationships in terms of collection of tables (entity types), each of which is defined in terms of columns (attributes) and rows (entities). Within this framework, a relationship is defined through the association of multiple entities. These relationships are specified via keys, which are the principal means for identifying entities within an entity type. This is illustrated in Figure B-12.

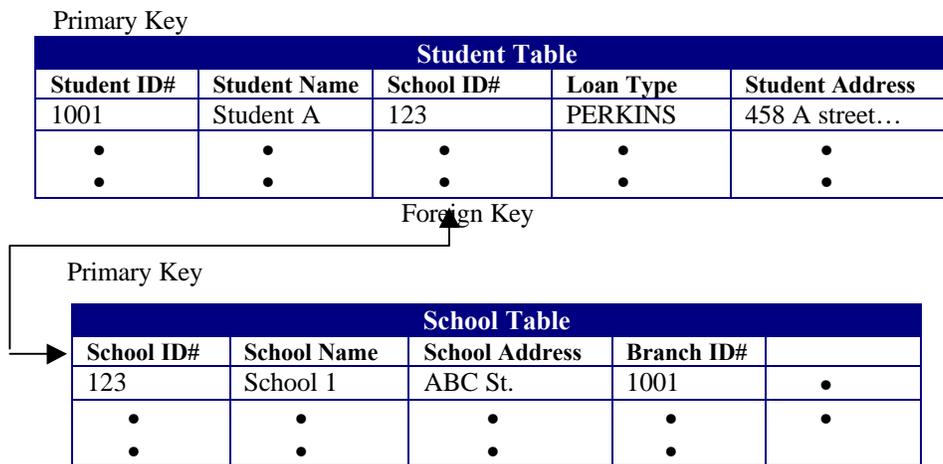


Figure B-12. Key relationships in a database

Today, all commercially available RDBMS products (or at least those with measurable market share) utilize some variant of the Structured Query Language (SQL) – a powerful, highly flexible, set-oriented language – for manipulating, defining, and controlling data. That is, SQL has become the predominant database management language for mainframe, minicomputer, enterprise & workgroup server, and even PC based RDBMS applications.

In addition to managing the control and execution of SQL commands, RDBMSs also manage data recovery, concurrency, security, and consistency – allowing multiple applications and users to share the database. Additionally, RDBMSs provide a variety of administrative and programmatic utilities and procedural extensions. Two such procedural extensions are stored procedures and triggers.

Stored Procedures are remote, compiled, and named procedures (functions) that are stored, accessed, and managed via the RDBMS. Stored procedures are typically used to enforce business rules and data integrity; however, the primary use of stored procedures is to implement “server-side” application logic within distributed system architectures. This is particularly the case in those architectures that are based on the two-tier logical software distribution strategy and use the Remote Presentation and Distributed Presentation physical distribution strategies, as described in Section 2. To use a stored procedure, the user (via the client application component) issues a remote function call, which invokes the stored procedure. This results in reduced network traffic and better performance because:

- Stored procedures, and the SQL statements therein, are precompiled.

- Only the client's function call and a fixed set of results traverse the network.

As illustrated in Figure B-13, these benefits are not available with “dynamic remote SQL” statements which must be:

- Compiled at run-time each time they are executed.
- Sent across the network – in their entirety – to the RDBMS for execution.

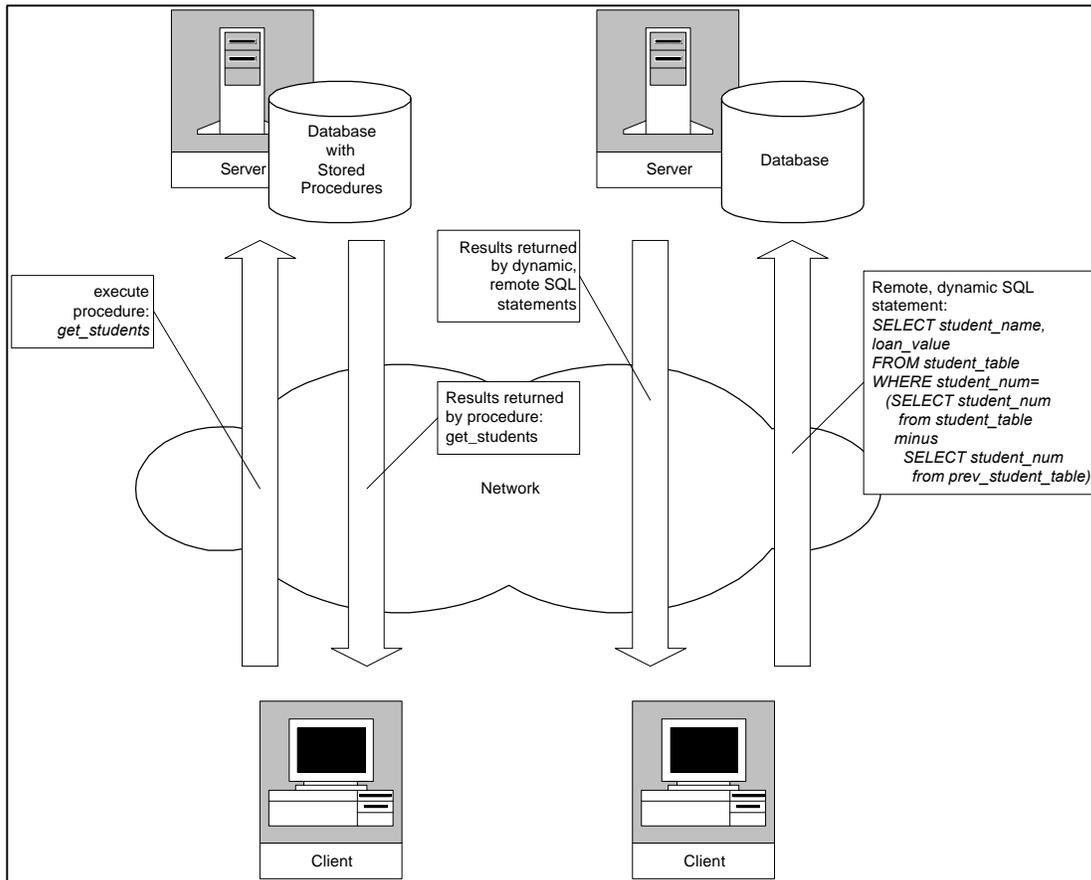


Figure B-13. Static and Dynamic Remote SQL Statement Execution

Triggers are implemented via stored procedures. However, triggers are automatically executed by the RDBMS in response to specific data-related events; whereas stored procedures are explicitly invoked by users (via client application components). Typically, triggers are associated with specific data update events – those executed via SQL DELETE, INSERT, and UPDATE instructions – and are used to implement “event responses” or “event handlers.” A separate event handler – trigger – can be defined for each type of SQL update event. Alternatively, triggers can be defined to handle any updates to a table. RDBMS triggers are typically implemented via proprietary SQL procedural extensions, which do not comply with the SQL-92 standard. As a result, specialized skill sets are required to develop RDBMS triggers. Additionally, much like stored procedures, triggers are not portable across heterogeneous RDBMS vendor products.

These extensions are useful; however, they are not standardized and, in many cases, can significantly reduce system flexibility, as stored procedures and triggers are not portable across vendor platforms and usually must be implemented via RDBMS proprietary software development environments and languages.

Nonetheless, faced with competing industry pressures for standardization and marketing pressures for product differentiation, these extensions have become useful tools that keep RDBMSs from becoming commodities, even as vendors embrace standards.

In addition to stored procedures and triggers, there are other features that differentiate commercially available RDBMS offerings. Some of these include:

- Concurrency Schemes** provide mechanisms that allow multiple transactions from multiple users to simultaneously access the data managed by the RDBMS. RDBMS offerings support various ANSI SQL-92 isolation levels, including: read-uncommitted (“dirty read”), read-committed, and repeatable-read. Read-uncommitted is lowest isolation level. When the RDBMS fetches a row, it does not place a lock and it does not respect any other locks. In the read-committed level, the RDBMS guarantees that all rows read have been committed. This level prevents reading data that is not committed and subsequently rolled back. Finally, the repeatable-read level asks the RDBMS to place a lock on every row that is fetched until the transaction is complete.
- Locking Strategies** provide a means of ensuring consistency in the data through the prohibition of certain forms of concurrency. Commercially available RDBMS adhere to disparate locking strategies – allowing data to be locked at the database, table, memory page, or row level. Several types of locks can be used to implement concurrency schemes, including binary locks and multiple-mode locks. Binary locks require data to be locked when reading and/or updating data within the database. In many situations, binary locks prove to be too restrictive. In these situations multiple-mode locks are often used. Multiple-mode locks allow data to be read- and write-locked. Read-locks are often referred to as “shared locks,” because they allow other transactions to read the locked item. Write-locks, on the other hand, are often referred to as “exclusive locks,” because a single transaction exclusively holds the lock on the item. The multiple-mode strategy increases data availability. Similarly, lock granularity – database, table, page, or row level – affect data accessibility, as illustrated in Figure B-14.

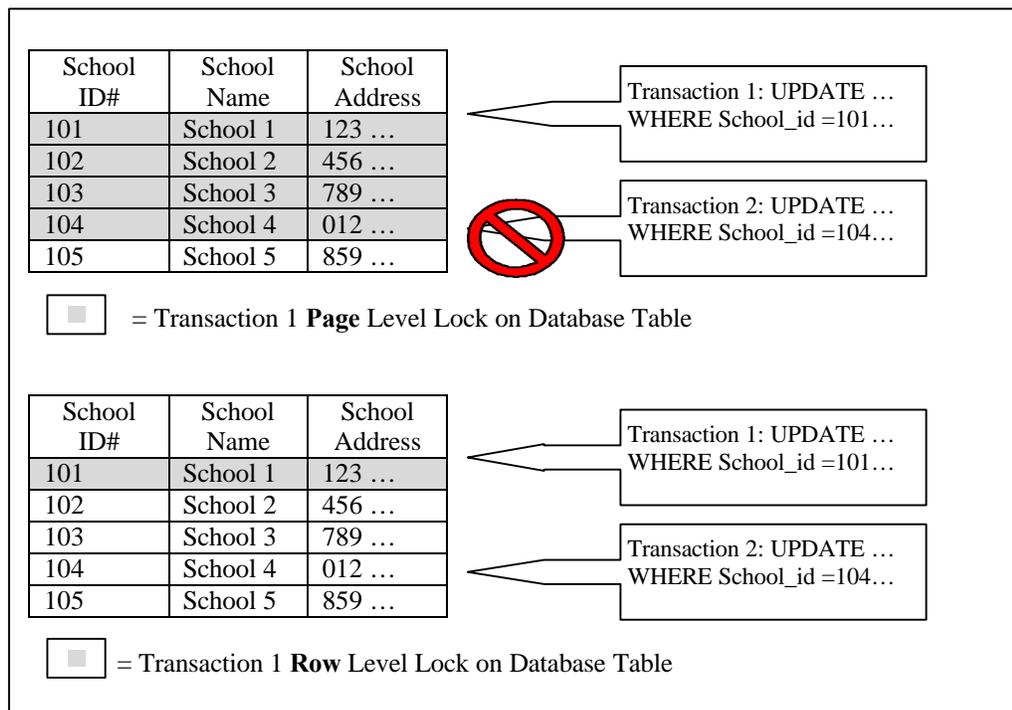
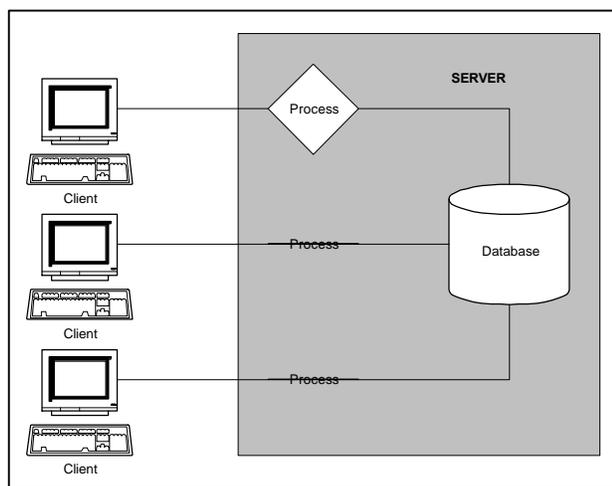


Figure B-14. Database Row Locking

- **Memory Management** facilities control how the RDBMS accesses and uses system memory and storage resources. RDBMS memory management strategies are only obviously different in one area -- user/client connection management. For client connections there are three widely followed memory management strategies. These strategies are:
 - **Process-Per-User Strategy** provides a separate process and address space for each client connection to the database. This effectively isolates client operations and increases fault tolerance. To implement this strategy, the RDBMS typically leverages the operating system's multitasking services. As a result, within symmetric multiprocessing environments, the RDBMS can rely on the operating system to transparently "load balance" client connections across the pool of available processors. The disadvantage of this strategy is that it consumes more system resources than other strategies and can be slower. As a result, although this strategy is very robust, it can also be performance constrained. This strategy is illustrated in Figure B-15.



**Figure B-15. Memory Management Strategies:
Process -Per-User**

- **Multi-Thread Strategy** does not rely on the operating system RDBMS related process management facilities. Rather, with this strategy the database server, connections, and other associated process "threads" are all executed within a single process address space. This is possible because the RDBMS provides its own process management and scheduling facilities. This strategy usually provides the best performance and requires the least amount of system resources (CPU cycles, memory, etc.). However, because process threads are sharing the same address space, a single erroneous thread can catastrophically affect all database services. This strategy is illustrated in Figure B-16.

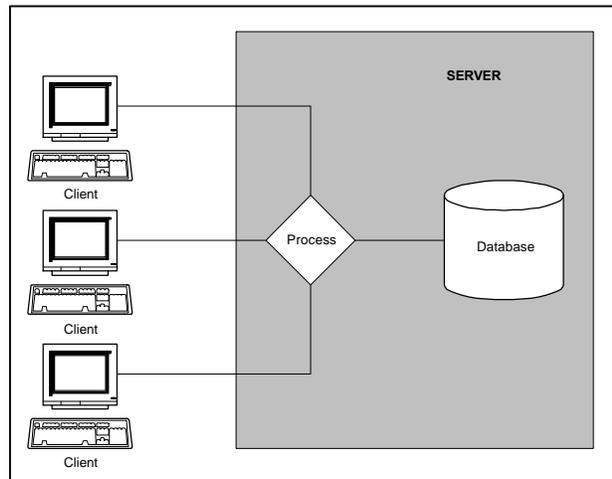


Figure B-16. Memory Management Strategies: Multi-Thread

- Queueing Strategy** provides process management service very much like those found in Online Transaction Processing (OLTP) monitors, such as AT&T's Top End and Transarc's Encina. That is, the RDBMS queuing strategy requires three components: 1) multi-thread client request "listeners," which "map" client connections (messages) to a queue manager; 2) the queue manager, which writes client messages to queues and delivers queued responses to clients; and 3) the pool of shared RDBMS processes, which read messages from the queue(s), process client requests, and write responses to the queue. This strategy provides the advantages found in the Process-Per-User strategy, without assigning a permanent process to each connection, which (much like the Multi-Thread Strategy) reduce processing overhead and resource requirements. However, queue latencies, which are associated with this strategy, can introduce performance constraints in systems supporting large numbers of concurrent client connections. This strategy is illustrated in Figure B-17.

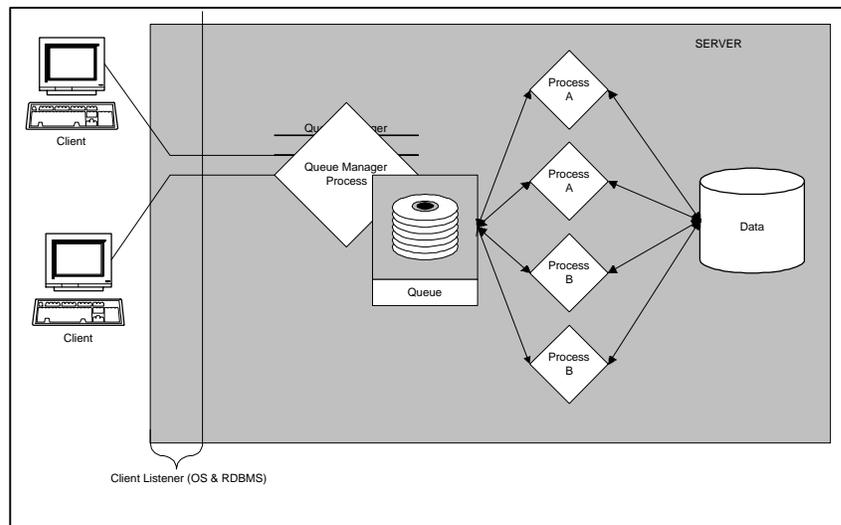


Figure B-17. Memory Management Strategies: Queuing

- **Parallel Processing** splits a task into smaller tasks that are executed on multiple processors simultaneously. It is designed to take advantage of symmetric multiprocessing hardware. Parallel processing can actively manage the use of hardware resources such as the CPUs and memory, maximize throughput, and resolve contention between OLTP and decision support applications. Parallel processing capabilities are widely used to improve database query processing performance.

Query processing normally involves 1) decomposing the query into multiple tasks, and 2) sequentially processing the multiple tasks – using the output of one task is a prerequisite input to the next task. Parallelization further divides query tasks into subtasks and facilitates the simultaneous processing of these subtasks. Parallelization comes in two forms: vertical and horizontal. Vertical parallelization refers to the degree of task completion required before a task can use the proceeding task’s results. For example, if a database scan task is to be followed by a join task (both subtasks to a query task), vertical parallelism enables the join to be started as soon as the data from the scan becomes available, not when the scan is complete. Horizontal parallelization, on the other hand, refers to the RDBMS’s ability to simultaneously process objects that are partitioned across multiple disks. For example, if a table that is partitioned across four disks needs to be scanned, the sequential RDBMS would read data off each disk in turn, while the parallel RDBMS would complete the task in a quarter of the time by reading all four disks at once.

- **Process Optimization** automatically determines which indices and what sorting algorithms to use when processing an SQL query. Some RDBMSs rely on rule-based optimization, which means that index usage is dependent on the way the SQL command is written. Cost-based optimizers, on the other hand, provide a more advanced optimization capabilities. A cost-based optimizer uses statistical data to analyze what data is in the database and what data is being requested. Additionally, cost-based optimizers use algorithms to determine the weighted cost of data retrieval alternatives and select the one that uses the least amount of resources.

Vendor	Product
Oracle Corp.	Oracle 8.0
IBM Corp.	DB2, Common Server DB2
Informix	Informix 7.1, Informix 8.0 XPS
Sybase	Sybase System 11, Sybase MPP
Microsoft	SQL Server

TECHNOLOGY: Data Warehousing

TECHNOLOGY DESCRIPTION:

When searching for information to support business decisions in today's dynamic global business environment, many business users find that the traditional sources of data – transaction-based systems – are inadequate in content, accessibility, form, performance, and availability. The problem often lies not with the data or their source, but in the limitations of current technology to bring together information from many disparate systems. Increasingly, the solution to these problems is data warehousing.

A data warehouse is an orderly and accessible repository of known facts and related data that can be used as a basis for making better management decisions. The data warehouse provides a unified repository of consistent data for decision-making that is subject-oriented, integrated, time-variant, non-volatile, accessible, transformed, and management-oriented. Bill Inmon, an important figure in popularizing the data warehouse concept, has defined these characteristics as follows:

- **Subject-Oriented:** Data is classified and organized around subjects that are meaningful to a organization. As illustrated in Figure B-18, these subjects are often defined by the critical success factors of the enterprise – its customers, its employees, its products, its suppliers – and other important subjects of management interest.

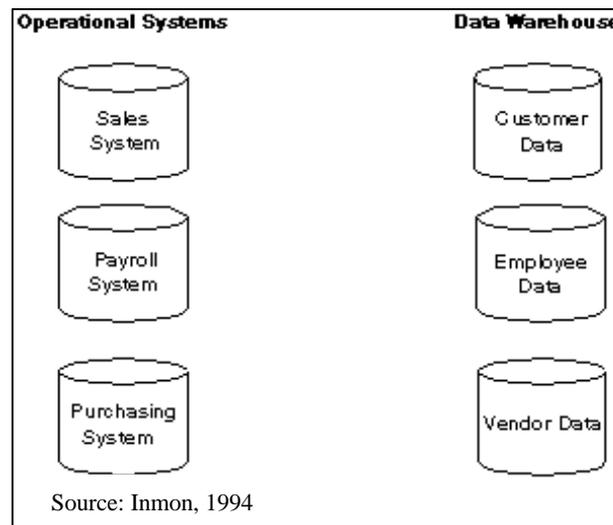


Figure B-18. Subject-Oriented Data Classification

- **Integrated:** The data in a data warehouse is pulled together from various operational systems and external data sources, enabling a cross-functional view of the enterprise. Integrated also refers to the fact that data definitions are standardized. For example, an order processing system may consider each location as a customer, whereas a marketing system may be looking at each enterprise as a customer. The process of integrating the data needs to resolve these differences between the two databases as the data are loaded into the data warehouse. This characteristic is illustrated in Figure B-19.

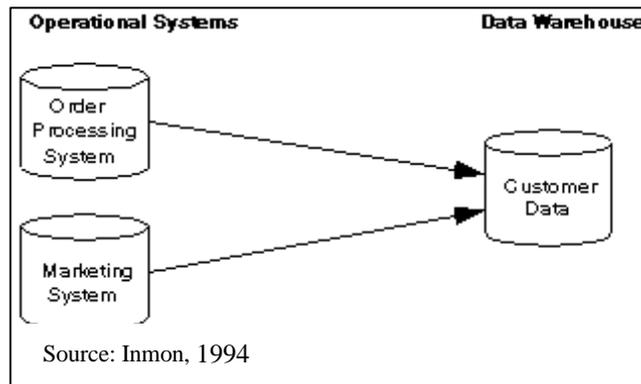


Figure B-19. Integrated Data Classification

- **Time-Variant:** A data warehouse contains both historical (5 years old to 10 years old) and nearly current data. The data in the warehouse are also stored in such a way that they support time-based analysis.
- **Non-Volatile:** The data warehouse is non-volatile because the data are sourced from one or more operational systems through a well-defined and controlled extraction and transformation process. Therefore, a data warehouse is comprised of snapshots of the data at specific moments in time: once per day, once per week, once per month. Also, the data warehouse is usually a separate, read-only database. This characteristic is illustrated in Figure B-20.

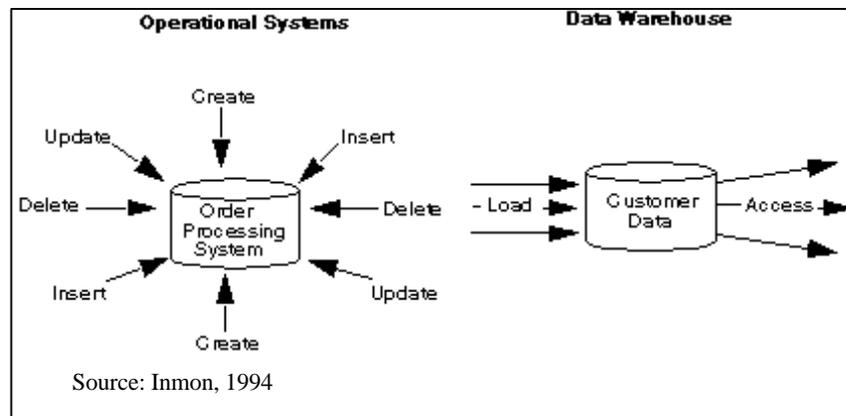


Figure B-20. Non-Volatile Data Classification

- **Accessible:** The data warehouse is primarily a distribution mechanism for the enterprise's data, so accessibility is critical. The data warehouse should provide simplified and timely access to data by making it easy to find what data are available, where they are, and how to access them.
- **Transformed:** For the data to be more accessible, they need to be transformed from the various operational system formats and external system formats into a single format. The data must be translated and summarized to make it consistent and easier to access and analyze.

- **Management-Oriented:** The primary purpose of a data warehouse is to aid in analysis and decision-making. Therefore, the data is organized to support decision support needs rather than operational needs. The data warehouse must be flexible and provide many ways to view and analyze the data that is relevant to users in a decision support and analysis environment, such as summary data and time-series data.

Today, data warehousing is considered the most effective way to transform "data" into "information" – providing critical repositories of timely and accurate information for decision-making. This information is increasingly important, as organizations need to adapt continually to changes resulting from competitive pressures, shrinking business cycles, a global market, and a transforming business environment.

The value of data warehousing lies in its ability to help users efficiently make well-informed decisions through analysis of the important organizational trends. As a result, users spend less time finding and accumulating data, and more time analyzing relevant information and working to implement solutions. That is, data warehousing provides management with access to the right information in the right format, at the right time.

To realize the benefits of data warehousing, data is extracted from operational systems and external information providers, then cleansed, aggregated, integrated, and transformed into a read-only database that is optimized for decision-making. That is, a data warehouse is a special-purpose database system where extracts of operational data are pre-processed (indexed, partitioned, and sometimes pre-aggregated) to improve query performance significantly. Once the data is appropriately stored in a data warehouse, it can be accessed and used through a wide range of access, analysis, and presentation software tools, including decision support systems (DSS), executive information systems (EIS), and analysis tools, such as data mining, statistical software, forecasting software, and simulation. (EIS, DSS, and data mining are described within a separate abstract.)

The concept of the data warehouse had its genesis in a 1988 International Business Machines Corp. research paper in the *IBM Systems Journal* entitled "An Architecture for a Business and Information System." IBM formally announced its Information Warehouse framework in September 1991. The stated objective was to provide a means for larger organizations to gain open access to data across their hardware platforms and vendor products (particularly, mainframe-based data, which can be difficult for users to access).

As illustrated in Figure B-21, data warehousing requires many technology components, which must work well together. That is data warehouse generation tools extract data from operational systems and external systems, transform data (cleanse and summarize), and load the data into the warehouse. Once in the warehouse, data is managed and used via data warehouse management and access tools. These warehousing tools are described as follows:

- **Data Warehouse Generation Tools** extract data from the operational databases, transform (or cleanse) the data, move the data to the server on which the data warehouse is located, and load the data into the data warehouse.

Preparing data to be loaded into a data warehouse involves extraction, transportation, and transformation. Extraction programs are periodically run to extract the relevant data from the operational systems. Customized programs are often used to do the data extracts, but utilities are now becoming more readily available to assist in this process. These programs specify the source operational system or external system and the extraction criteria. In order to prepare the data for the data warehouse, it must be:

- Filtered to eliminate unnecessary details or fields.
- Cleansed to eliminate incorrect or duplicate data.
- Converted and translated into the warehouse database format.
- Consolidated and aggregated from multiple sources.

Cleaning the data is especially important and difficult. Valuable data, such as customer or mailing data, must be accurate, but it is often difficult to recognize that multiple entries represent, in fact, the same entity (such as Bill Jones -- 123 Willow Street, William Jones -- 123 Willow St., and W. Jones -- 123 Willow Street). Additionally, data that is stale, redundant, or of poor quality must be identified and fixed before being introduced to the data warehouse.

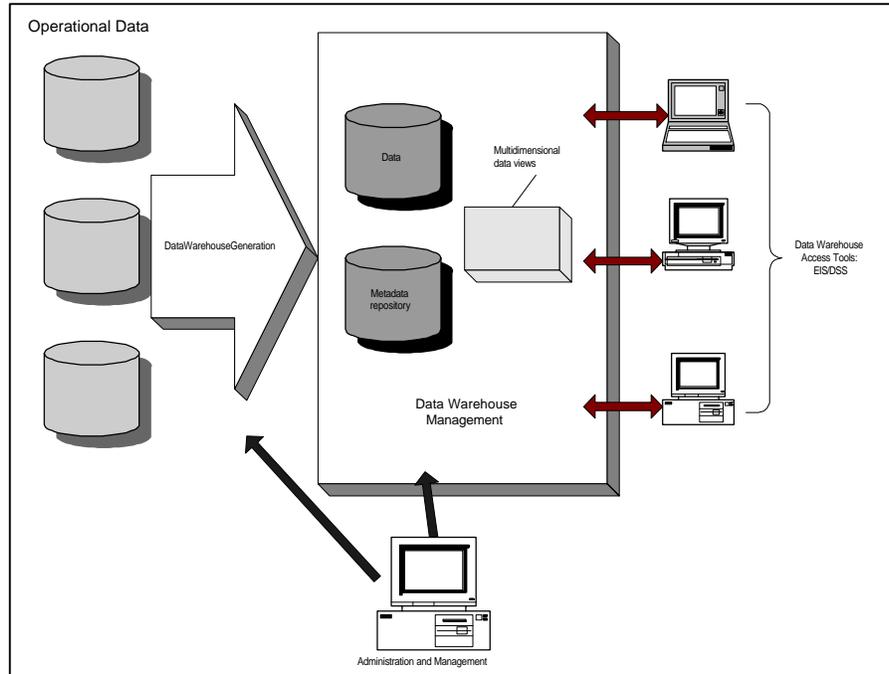


Figure B-21. A Data Warehouse Framework

Sophisticated data “cleansing” tools have emerged to help improve the quality of data before it is loaded into a data warehouse. These tools typically use pattern analysis, fuzzy logic, lexical analysis, and statistical matching to identify and consolidate logically redundant data such as names and addresses. In addition to cleansing data, many of these tools also facilitate access to data stored within heterogeneous legacy file formats and data structures.

After the data have been transformed into the appropriate format, it is loaded into the data warehouse. During the loading process the data is properly organized for easy access. This process may include summarizing the data, calculating derived values, denormalizing the data, time stamping the data, and building appropriate indices.

Data for warehouse-based applications is organized quite differently than data for online transaction processing (OLTP) applications. Typically, warehouse data is heavily indexed to improve data access performance (since there are few, if any, on-line updates of the data). OLTP data, on the other hand, has fewer indices because of the extra overhead associated with updating all the indices when data is modified. Additionally, warehouse data is often highly summarized and pre-calculated to provide faster performance for queries (in fact, it is one of the most effective ways of improving performance), whereas OLTP data are rarely summarized because of potential data integrity problems when updating the data.

Warehouse data is also denormalized (typically, fewer but larger tables) to reduce the number of joins necessary when querying the data to improve performance, whereas OLTP data is normalized to improve data integrity for updates. Another major difference in data organization is the use of data partitioning: OLAP applications use data partitioning (breaking up a big table into

several smaller tables, such as January, February, and so on, instead of an annual one) to speed data loading and improve data access, whereas OLTP applications rarely use this form of partitioning because of data integrity concerns associated with updating the data.

Data Warehouse Management Tools are typically based on parallel database management system and multidimensional database technologies and are used to help manage operations of the data warehouse throughout its operational life cycle. (Parallel processing DBMS capabilities are described within the RDBMS abstract and multidimensional data is described within the EIS/DSS abstract.) These operations include data quality assurance, systems management, performance management, and security (security is of particular importance, as data warehouses, by design, make data easier to understand and access). However, the most important service provided by data warehouse management tools is database management.

The data warehouse database stores, manages, and stages data for end-user access. It is the core component of a data warehouse and is often referred to as an online analytical processing (OLAP) database.

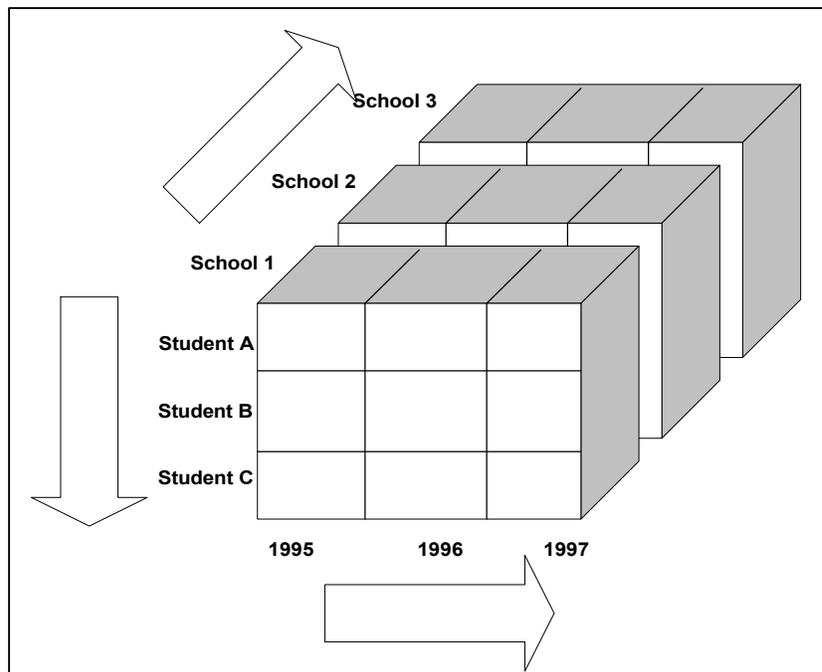


Figure B-22. Multidimensional Data Cube

As Figure B-22 illustrates, an OLAP database can be visualized as a multidimensional cube, where the dimensions represent business data organization. For example, using an OLAP database, the Education Department could review data via multiple dimensions such as time, student, school, and disbursement.

Like a spreadsheet, an OLAP database stores related data in blocks or cells. These blocks inside the cube are where the measures of the business are archived. For example, a block in the 3-D cube could be defined as the disbursement of a specific loan, to a student at a specific school, on a certain date (time). As a result of this organization, OLAP databases allow users to "slice-and-dice" along each of the dimensions of the data and "drill down" or "roll-up" several layers of consolidated data.

OLAP databases are typically implemented via multidimensional database technologies (also known as MOLAP, for Multidimensional OLAP) or via analytical engines, which provide virtual

multidimensional views from data stored with relational database technologies (commonly referred to as ROLAP, for Relational OLAP). However, despite these implementation differences, the generalization can be made that database products must include multidimensional functionality in order to support OLAP.

The key attraction of an OLAP database is its design, which facilitates intuitive data navigation, analysis, and presentation. In addition, OLAP databases also provide the ability to:

- Define aggregation hierarchies and interrogate all aggregation levels at any dimensional intersection.
 - Built-in analytical and computational features such as roll-up and drill-down capabilities.
 - Deliver improved/optimized complex ad hoc query performance.
- **Data Warehouse Access Tools** provide non-technical users with access the data stored in the data warehouse. The ease-of-use and capabilities of these tools are key determinants of the user's perception of the value and success of the data warehouse. These access tools can support predefined and ad hoc data access, data analysis, and data presentation, and increasingly, data mining. Data access tools EIS and DSS solutions. These technologies are described within the Executive Information System/Decision Support System abstract.

COMMERCIAL OFFERINGS:

Vendor	Product	Product Type
Vality Technology Inc.	Integrity Data Reengineering Tool	Data Warehouse Generation
IBM Inc.	Visual Warehouse, Data Guide, Visual Query	Data Warehouse Generation, Management, and Access
Prism Solutions Inc.	Prism Warehouse Manager	Data Warehouse Generation
Oracle Corp.	Oracle Express Server	Data Warehouse Management
MicroStrategy Inc.	DSS Server	Data Warehouse Management
Informix Software Inc.	Informix-Online XPS	Data Warehouse Management and Access
Red Brick System Inc.	Red Brick Warehouse VPT	Data Warehouse Generation and Management
Apertus Technologies Inc.	Enterprise/Integrator	Data Warehouse Generation
Information Builders Inc.	Enterprise Copy Manager	Data Warehouse Generation, and Management
Platinum Technology Inc.	InfoRefiner, InfoBeacon	Data Warehouse Generation and Management

TECHNOLOGY: Operating Systems

TECHNOLOGY DESCRIPTION:

All programmable computing devices, from workstations to mainframes, use an operating system. However, all operating systems are not created equal. As a result, the scalability and robustness of supported services tend to increase when moving from the desktop to a server, and again when moving to a server/host. In effect, a desktop operating system offers a subset of the functionality provided by a server operating system, and an even smaller subset of the features found in a server/host operating system. This is largely the case because server/host operating systems are generally based on mature technologies and were originally designed to simultaneously support thousands of users and transactions. By contrast, traditional desktop and server operating systems typically support much smaller work groups and are generally less mature and therefore less stable than server/host operating system products.

The following paragraphs describe the following server and server/host operating systems:

- UNIX.
- Microsoft's Windows NT Server.
- IBM's MVS/OS390.
- IBM's VM/ESA.
- IBM's OS/400.

UNIX is a multi-user operating system that was written in the late 1960s at AT&T Bell Laboratories. UNIX was written in C, a portable, processor-independent language. There are versions of UNIX available for Digital Alpha, HP-PA RISC, Intel, Mips, PowerPC, and Sun SPARC microprocessors, among others.

In early 1993, Novell acquired the rights to UNIX's source code from AT&T in an attempt to unify the UNIX market and offer a single version of UNIX. This acquisition was widely considered a preemptive strike by Novell against Microsoft's Windows NT, which was to ship in the summer of 1993. However, Novell failed to unify the UNIX market because UNIX vendors declined to license and sell Novell's UNIX product, called UnixWare.

Novell finally gave up its unification efforts in 1995 by transitioning the UNIX trademark to the X/Open Co. Ltd. (now The Open Group) and selling UnixWare to SCO. Therefore, any operating system that meets the requirements of The Open Group can now be branded a UNIX operating system.

Many companies have licensed the rights to UNIX and added proprietary extensions to it. As a result, the UNIX market is fragmented so that one vendor's version of UNIX might not be compatible with another vendor's version of UNIX.

All versions of UNIX support built-in TCP/IP networking, network management, directory services, multi-user/multitasking services, and interprocess communication.

UNIX is a popular operating system for running enterprise databases and for Internet applications. The sample versions or "flavors" listed below are derived from the AT&T source code and are widely installed within corporations:

Digital UNIX -- Formerly OSF/1, Digital UNIX runs on Digital's RISC-based Alpha workstations and servers. Digital UNIX supports 64-bit computing and is among the first UNIX versions to support a specification that aims to let UNIX applications run on any version of UNIX. Digital UNIX is based on UNIX System V Version 4.0.

HP-UX -- HP-UX runs on HP's PA-RISC processor, used in HP 9000 workstations and servers. HP-UX is based on UNIX System V Version 3.2. HP is working with SCO to

enhance HP-UX to run on a next-generation, 64-bit processor under joint development by HP and Intel.

IBM AIX (Advanced Interactive eXecutive) -- AIX runs on IBM's PowerPC-based RS/6000 workstations and servers, ES/9000 mainframes, and PCs. It is based on UNIX System V Version 3.2.

SCO OpenServer/SCO UnixWare -- Formerly called SCO UNIX, OpenServer is the leading version of UNIX for Intel hardware. OpenServer is based on UNIX System V Version 3.2. Acquired from Novell in late 1995, UnixWare is based on UNIX System V Version 4.2. SCO is working to merge OpenServer and UnixWare into a single offering, although it is unclear when that work will be completed.

Silicon Graphics' IRIX -- Based on UNIX System V Version 4.0, IRIX is particularly popular for imaging and intensive graphics uses, including special effects in commercial movies, and supports Indigo Magic, the company's desktop product for visual computing.

Sun's Solaris -- Solaris is available for Sun's SPARC and 80x86 computers. A version of Solaris is also under development for the PowerPC. Solaris is based on a merger of UNIX System V Version 4.0 and SunOS.

Although thousands of applications are available for UNIX, it remains difficult for developers to write a single application that can run on multiple versions of UNIX – largely because no two UNIX versions are exactly alike. The Single UNIX Specification (formerly known as Spec 1170) was supposed to give software developers a single set of APIs that were supported by every major version of UNIX, thereby unifying UNIX offerings. However, UNIX vendors have been slow to offer support for the Single UNIX Specification. In effect, the goal of creating a market for shrink-wrapped UNIX applications remains largely unfulfilled.

Microsoft's Windows NT Server (formerly known as Windows NT Advanced Server) was originally released in 1993 and offers many of the features offered by UNIX operating system. Like UNIX, Windows NT was developed in portable C code. As a result, Windows NT Server is available for Digital Alpha, Intel, and PowerPC microprocessors. (Test versions of Windows NT Server for HP/PA RISC and Sun SPARC microprocessors have also been developed, although to date, they have not been released.) Windows NT Server is a 32-bit operating system that supports symmetric multiprocessing and enforces security access controls on the file system and other objects.

Windows NT Server did not generate significant sales until 1994, with the release of the second major version, Windows NT Server 3.5. Windows NT Server 3.5 included a new TCP/IP stack, and gained a groundswell of server application support from Microsoft, Lotus Development Corp. (now an IBM subsidiary), Computer Associates (CA) International Inc., and other server software developers. In late 1996, Microsoft delivered Windows NT Server 4.0. It included support for the Windows 95 interface, cryptography APIs, improved scalability on multiprocessor servers, and the Internet Information Server (IIS) – integrated Web server software.

Windows NT Server 4.0 is still not a full-function server operating system. Nonetheless, many IT organizations are deploying Windows NT Server to run departmental databases, Internet/intranet servers, and support file and print services – largely because cost of ownership is less than with alternative server operating system solutions. Many UNIX application developers, including IBM, Oracle, Sybase, and Informix, are now offering applications for Windows NT Server. Microsoft itself offers BackOffice, a suite of software for Windows NT Server that includes the SQL Server database, Systems Management Server for PC administration, SNA Server for mainframe connectivity, and Exchange Server for messaging services. Web servers for NT are also widely available.

Like Windows NT Workstation, Windows NT Server is available for Intel, Digital Alpha, and PowerPC microprocessors. However, software developers must recompile their applications from the original

platform for use on additional platforms. Because most companies run Windows NT Server on Intel servers (and sometimes, Digital Alpha servers), few vendors have ported their Windows NT Server applications to other platforms.

Many enterprise hardware and software providers, such as Tandem, are seeking to diversify their revenue streams by adding value to Windows NT Server. In May 1996, Tandem announced that it will extend its ServerNet clustering technology, NonStop transaction processing technology, and NonStop SQL database technologies to Windows NT Server. Announcements such as Tandem's and those made during 1996 by CA, Digital, SAP AG, and other enterprise players, ensure that Windows NT Server will be an increasingly attractive platform for higher-end, mission-critical applications.

On May 20, 1997, Microsoft announced an enterprise edition of the Windows NT Server 4.0 operating system, which will support up to eight processors. Additionally, the enterprise edition of NT will ship with Microsoft's new Transaction Server and Message Queue Server. In an effort to demonstrate NT's ability to support "higher-end, mission-critical" applications, Microsoft also demonstrated these technologies. Specifically, using a mock application of a global bank's automated teller machine (ATM) network, 20 Compaq servers, each running Microsoft SQL Server, were able to handle nearly 14,000 transactions per second, or more than 1.2 billion transactions per day if sustained over 24 hours. The setup included five servers running Microsoft Transaction Server, which distributed the ATM transactions among machines. Additionally, in a second demo, Microsoft unveiled a terabyte-size database, loaded with satellite images of the earth, running on a four-processor Digital Equipment AlphaServer 4100 running Microsoft's NT and SQL Server 6.5, Enterprise Edition.

Multiple Virtual Storage (MVS) and Virtual Machine (VM) from IBM dominate the mainframe operating system world. MVS was introduced in 1974, and has since been continuously enhanced to support on-line applications such as CICS (Customer Information Control System). Additionally, MVS will run on mainframes built by Amdahl Corp., Fujitsu Ltd., and Hitachi Data Systems Corp.

MVS is designed to manage very large systems with multiple terabytes of data. MVS/SP4 (System Product 4) provides support for the Open Software Foundation's Distributed Computing Environment (DCE) services, which offer distributed system infrastructure features, such as directory and security services.

IBM's MVS/ESA (Enterprise Systems Architecture) current offering – OpenEdition MVS – received UNIX certification from The Open Group in 1996. It adds UNIX APIs that help transform the MVS control program into a super-UNIX-like environment. Integrating the APIs into the MVS base allows IBM to address several of the shortcomings of native UNIX: robustness, integrity, and diverse workload handling.

IBM's flagship version of MVS is now OS/390 Release 2. OS/390 bundles the core MVS/ESA operating system with applications, enabling components and communication functions from more than 50 products that formerly were sold separately. Through OS/390, IBM has reduced user-testing requirements because the software suite is pre-bundled and tested prior to shipment. In addition, competitive pricing makes OS/390 an attractive choice. The Gartner Group estimates that perhaps 50 percent of the MVS installed base will be using OS/390 by the end of 1998.

OS/390 also currently includes a communications server and is available with an optional security server. The optional OS/390 Security Server is a full-function security service that supports RACF (Resource Access Control Facility, an access and control system that authorizes access to resources and logs unauthorized access attempts and accesses to protected data sets).

For additional reliability and scalability, OS/390 supports Parallel Sysplex, a technology that couples multiple S/390 servers into a single, logical server. Using Parallel Sysplex, workloads can be distributed dynamically to any available processor.

IBM has announced that it intends to deliver a new release of OS/390 approximately every 6 months to improve customer planning productivity and efficiency. Specifically, IBM is transforming OS/390 into an open operating system that supports the following:

Internet services – In April 1996, IBM announced the OS/390 Internet Bonus-Pak, a free software package that lets customers use their mainframes as Web server platforms. Several airlines, major educational institutions, and global hotel chains have already deployed IBM's new Web server software on their IBM mainframes because it can support more than 14,000 concurrent user requests.

UNIX standards – In September 1996, IBM followed its Web announcement by revealing that OS/390 had achieved XPG4 UNIX 95 Profile Brand (UNIX 95) certification. According to IBM, OS/390 now provides customers with full UNIX capabilities built directly into the operating system, and is the first server/host operating system to offer such support for UNIX.

Windows NT applications – OS/390 will run Wind/U, a third-party product from Bristol Technology Inc. that allows developers to compile and execute Microsoft Windows applications on OSF/Motif.

IBM's VM/ESA (Virtual Machine/Enterprise Systems Architecture) manages a computer system so that all of its resources – processors, storage, and I/O devices – are available to many users simultaneously. VM is a multiple-access system that contains three major elements:

Control Program (CP) – A program that controls the resources of the real computer to provide multiple virtual machines. Each virtual machine can run a different operating system and can provide virtual storage support for operating systems that do not offer such support.

Conversational Monitor System (CMS) – A component that gives users a wide range of conversational facilities, including creation and management of files and compilation, testing, and execution of programs. CMS supports office productivity tools, software development, database management, decision-support tools, and a large library of third-party software.

Remote Spooling Communications Subsystem (RSCS) – A component that enables users to transmit files to and receive files from remote stations in the Remote Spooling Control System (RSCS) teleprocessing network.

The latest VM/ESA release, known as VM/ESA Version 2 Release 2.0, is POSIX-compliant and provides the following:

A smooth transition into the next century. VM/ESA Version 2 Release 2.0 is the only VM/ESA release with support for Year 2000 issues. That is, it resolves dates with two-digit years with correct century information, and enables testing of MVS, OS/390, VM, and VSE systems with Year 2000 dates.

Enterprise-class performance, due to its ability to run on IBM's S/390 Multiprise 2000 and S/390 G3 Enterprise Server platforms.

Enhanced network communications via IBM's TCP/IP and Web third-party server products.

OS/400 is IBM's operating system for the AS/400 line of computers. The AS/400 has sold 400,000 units since its introduction in 1988, and nearly 28,000 OS/400 applications are available worldwide, according to IBM, including 3,000 client/server applications such as SAP R/3 and PeopleSoft.

OS/400 Version 3 Release 1 includes OSF's DCE services including authentication, directory services, and remote procedure calls. OS/400 supports an Integrated Language Environment (ILE) C/400 programming language, which enables applications to perform two times to three times faster compared with previous C applications written for the OS/400. It also includes native ODBC support, allowing distributed client applications to access data from OS/400 data storage.

In June 1996, IBM unveiled its latest upgrade to OS/400, known as OS/400 Version 3 Release 2. This new OS/400 release supports Internet access for AS/400-connected users, enhanced security features for Internet access, and integrated Lotus Notes support.

IBM for the first time is positioning AS/400 and OS/400 to compete against PCs on price. For instance, IBM offered a special \$7,995 AS/400 starter kit that runs PC software as well as AS/400 applications. IBM also demonstrated a thin-client solution called Network Station, a network computer with a graphical interface designed for network-centric computing that will cost around \$700 and work with the AS/400. By contrast, Pentium Pro servers running Windows NT and PC applications typically sell for \$10,000 to \$15,000 or more, and the PCs linked to them have an annual cost of ownership of \$5,000 or more. With an \$8,000 AS/400 and a \$1,000 thin client, IBM hopes to convince customers that the AS/400 and its OS/400 operating system offers a lower-cost alternative to networks comprising Pentium Pro servers and Pentium desktops.

COMMERCIAL OFFERINGS:

Vendor	Product
Sun Microsystems	Solaris
Hewlett Packard	HP-UX
Digital	Digital UNIX
Silicon Graphics	IRIX
SCO	OpenServer, SCO UnixWare
Microsoft	Windows NT Server
IBM Corp.	AIX, MVS/OS/390, VM/ESA, OS/400

TECHNOLOGY: Electronic Data Interchange

TECHNOLOGY DESCRIPTION:

Electronic data interchange (EDI), defined as "the transmission of business transaction information in computer-readable form between organizations in a standard format," is an electronic means to improve the quality and availability of business information. That is, EDI's provides direct electronic transmission capabilities that facilitate business transaction efficiencies and data integrity improvements – this allows employees to focus on business issues instead of data processing and error correction.

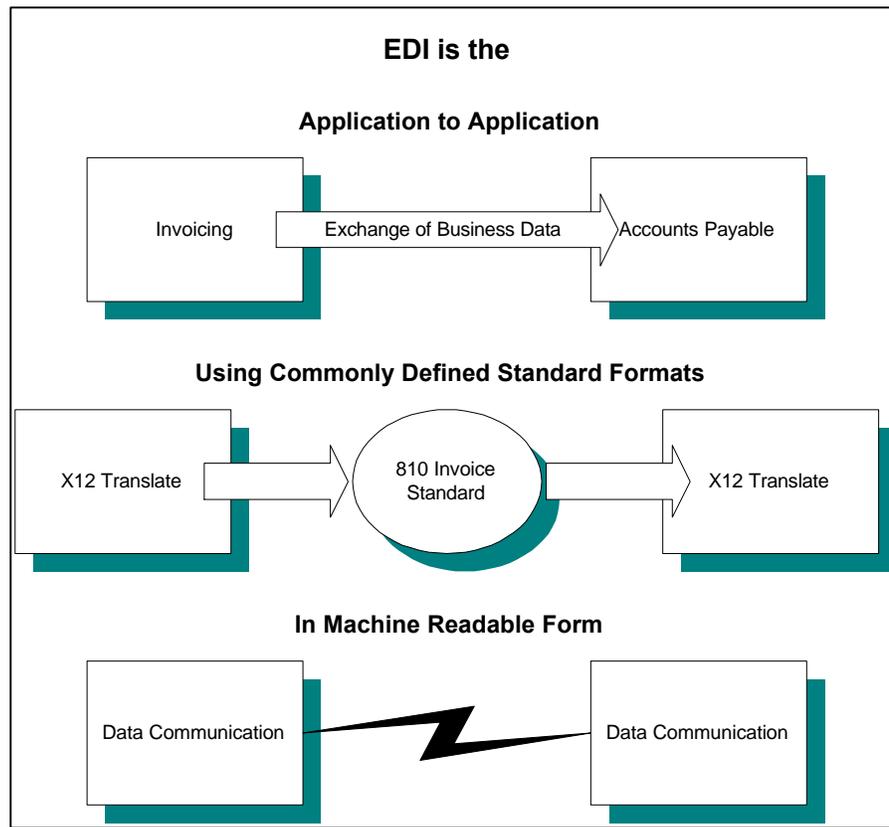


Figure B-23. What is EDI?

EDI is facilitated by a technology infrastructure that permits routine information exchanges between computer-based processes. Processes which exchange information through EDI are typically tightly coupled business applications owned by two or more trading partners. Examples of tightly coupled processes include:

- A school's financial aid system, which generates disbursement requests (invoices) for transmission to the EASI/ED target system.
- A vendor's billing system, which prepares invoices for transmission to a customer's accounts payable system.
-
- A customer's purchasing system, which generates orders for direct transmission to a vendor's sales order entry system.

Trading partners who exchange large volumes of transactions have discovered that paper-based exchanges are inherently inefficient and human interfaces are expensive and sometimes unreliable. With these deficiencies in mind, in 1979 several industry groups and individual companies, led by the Credit Research Foundation (CRF), petitioned the American National Standards Institute (ANSI) to accredit a standards committee to address this need. As a result, the Accredited Standards Committee X12 (ASC X12) was authorized by ANSI later that year.

ASC X12 was established with an open membership. Leveraging the accomplishments of other Electronic Business Data Interchange initiatives, the committee enlisted the participation of a wide cross-section of American businesses in the development of a broad dictionary of data elements, transaction types, and syntax/design rules. Today more than one hundred different types of transactions can be exchanged via the technique that has come to be known as EDI. ASC X12 estimates that 14,000 organizations use the EDI standard and, as a result, enjoy benefits such as:

- Reduction of paperwork and associated savings
 - One-time data entry
 - Reduced errors, improved error detection
 - Higher productivity without increasing staff
 - Reduced clerical workload
 - On-line data storage
 - Reduced postage and handling costs
 - Reduced printing costs of forms

- More timely communications
 - Rapid exchange of data
 - Reduced mail/delivery time
 - Increased customer service quality

- Uniform communications with trading partners

One primary function of EDI software is to transform data to and from a defined EDI standard format (e.g., ANSI ASC X12 standard). This computer-based process is appropriately referred to as the "translation" function.

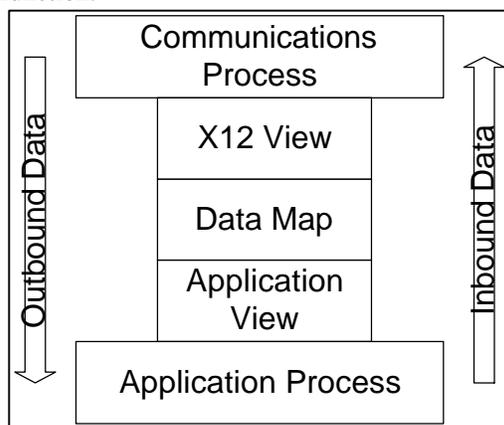


Figure B-24. EDI Translation Process

EDI begins and ends with business applications, which share data, but have different methods of viewing and processing this data. For example, a school's financial aid system may use data, such as social security number, loan/grant amount, and school name, to request or invoice the disbursement of funds on behalf of a student. These data elements will be received by the EASI/ED target system, where they may be used to authorize funding and request payment.

While exactly the same data values are used by both the school and ED, the semantics, syntax, and technologies used by the communicating systems may be quite different – largely because dissimilar processes mandate disparate technologies. To solve complications associated with the heterogeneity of trading partner systems, EDI

uses a set of commonly defined transaction formats – transaction sets – to exchange data. A partial listing of these X12 transaction sets is provided in Figure B-25.

When using EDI, the sender's originating application – for example, the school financial aid system described above – produces the same data regardless of the delivery mechanism. That is, the business

functions supported by the application do not change based on whether EDI is being used or not. Simply put, with EDI, the disbursement request data is sent to the translation process software instead of the form printer, but the services delivered by the application do not change – although they will probably be delivered more efficiently.

The translator software accepts the student financial aid system's view of the data, applies a user-defined map and formats the disbursement request into a standardized EDI transaction (for example, transaction Set 810). The transaction is delivered through a communications process to ED, the receiving organization, where the EDI process is reversed. For example, an ASC X12 standard invoice may be filtered through a map defined by ED and converted into a view of the data required by the EASI/ED target system. Once the EDI transaction is received, the information therein is processed the information just as though it had originated from a paper disbursement request or invoice.

TRANSACTION SET ID	STANDARD TITLE	X12 REFERENCE NUMBER
130	Student Educational Record (Transcript)	X12.89
131	Student Educational Record (Transcript) Acknowledgment	X12.90
135	Student Loan Application	X12.198
139	Student Loan Guarantee Result	X12.265
144	Student Loan Transfer and Status Verification	X12.94
146	Request for Student Educational Record (Transcript)	X12.121
188	Educational Course Inventory	X12.322
189	Application for Admission to Educational Institutions	X12.321
190	Student Enrollment Verification	X12.264
191	Student Loan Pre-Claims and Claims	X12.276
194	Grant or Assistance Application	X12.372
198	Loan Verification Information	X12.359
810	Invoice	X12.2
820	Payment Order/Remittance Advice	X12.4

Figure B-25. EDI X12 Transaction Set

Translation software packages with varying degrees of sophistication exist for virtually every computer platform. Many large organizations use translators running on mainframes or enterprise servers that centrally support all of the organization's business application systems. As applications are selected for EDI support, new maps and application data definitions are added to the central translator's libraries. These maps are used to prepared outgoing application data for X12 formatting and incoming X12 transactions for application use. Creating maps and profiles replaces the more time-consuming activity of programming individual interfaces for each trading partner's view of each target application. Additionally, many translation products maintain partner and communication profile tables, which direct the output of the translator to the correct delivery mechanism.

In addition to the translator software, communication alternatives must be considered when implementing EDI solutions. Many EDI operations are conducted using what is called "point-to-point" connectivity. Point-to-point connectivity means that the connection between sender and receiver is fixed and used solely for the purpose of EDI.

How long the points are connected depends on whether the EDI user selects dial-up or dedicated mode. In dial-up mode, the connection is established by placing a call through the public switched telephone network. As long as the computers at opposite ends of the telephone circuit are exchanging data the connection will remain open. The connection ends when one of the computers terminates the call by hanging up. In dedicated mode, a telephone circuit is connected constantly. The computers connected to either end of the circuit may exchange data at any time by entering into a predefined dialogue. No dialing is required since the telephone circuit is always available.

The dedicated mode level of service is generally not required as EDI exchanges tend to occur in bursts of file transfer activity at the beginning and end of a company's batch production cycles. A dedicated circuit is needed when transactions are volume or time-sensitive, as is the case with just-in-time programs. In addition to dedicated or dial-up mode connectivity, an EDI user must decide whether to support direct communications with each trading partner or engage a Value Added Network (VAN) provider. For a fee, the VAN will collect and disseminate a firm's EDI transactions.

A VAN customer supports a single connection between itself and the VAN's nearest point-of-presence (usually a local telephone call) in dial-up mode. The VAN services senders by delivering standard EDI transactions (such as ANSI ASC X12 transaction messages) to the designated receivers. The sender may communicate with the VAN at its convenience and the VAN will deliver the sender's transactions to what are called Electronic Mailboxes. Each mailbox belongs to a specific recipient of transaction data and transactions from multiple senders may be collected in the same mailbox.

At its convenience, the receiver of EDI data from a VAN establishes a communications session and requests delivery of all transactions stored in its mailbox. Since the sender's and receiver's communications sessions are independent of one another, the VAN can facilitate asynchronous processing – removing the requirement that trading partner systems be simultaneously available. The fact that sender and receiver are never directly connected also means that a VAN user is not required to support various data communications protocols and speeds based on the abilities of individual trading partners.

COMMERCIAL OFFERINGS:

Vendor	Product
Amsys North America Inc.	HostBridge for DEC/EDI
CMI-Competitive Solutions, Inc.	TRANS4M EDI
Data Management Strategies, Ltd.	Pro_EDI
Datacom Global Corp.	EDI-Answer®
GE Information Services	A-EDI Application Integrator
Frontec AMT, Inc.	AMTriX™
IBM Corp.	DataInterchange
Momentum Systems Limited	Intelligent Network Gateway
Digital Equipment Corp.	DEC/EDI Software
Sterling Software	GENTRAN
EDI Support, Inc.	A-Translation
Extol, Inc.	Extol EDI

TECHNOLOGY: Networking Infrastructure Technologies

TECHNOLOGY DESCRIPTION:

Network technology has evolved to support communications not only among coworkers, but also with suppliers, clients, and trading partners. As a result, networks are increasingly distributed, complex, pervasive, and heterogeneous. With these complications in mind, the Open Systems Interconnection (OSI) model was developed.

As illustrated in Figure B-26, the OSI model provides a framework for defining standards that can be used to “link” heterogeneous systems. That is, OSI is intended to facilitate communication and information exchange through any standardized communication facility executing standardized OSI protocols.

The OSI model is composed of communication functions, which are partitioned into a hierarchical set of layers. Each of these layers perform a subset of the functions required for system communication. Within this hierarchical construct each layer relies on the next layer to perform more primitive functions and to conceal the complexity of those functions.

This abstract defines technologies that provide services consistent with the Data Link layer of the OSI model. That is, this abstract describes technologies that organize data into a set of frames and supplement each data frame with control data, which is used to facilitate reliable transmission across the physical network link.

Data Link technologies are specifically considered within this abstract because data transfer rates⁶, network costs, topology, and media options are significantly influenced by Data Link protocol decisions. Specific technologies described within this abstract include:

- Ethernet and Fast Ethernet.
- Token Ring.
- Fiber Distributed Data Interface (FDDI).
- Asynchronous Transfer Mode (ATM).
- Frame Relay.

It should be noted that some of the technologies described within this abstract are not strictly classified as Data Link Protocols (for example: frame relay). These technologies have been included because they provide Data Link services.

Ethernet is the most popular data link protocol in use today and has become the most common way to implement LANs. Commonly referred to as “Shared Ethernet,” this protocol typically provides 10Mbps data rates and utilizes Carrier Sense Multiple Access with Collision Detection (CSMA\CD) technology. With CSMA\CD any station is allowed to transmit data at any time, as long as the network is not occupied by transmissions from other stations. When two or more stations transmit data simultaneously, a collision

⁶ Data rate is the measure of the speed at which data that can be transferred over a medium. Data rates are typically articulated in terms of megabits per second (Mbps) or kilobits per second (Kbps).

OSI Architecture

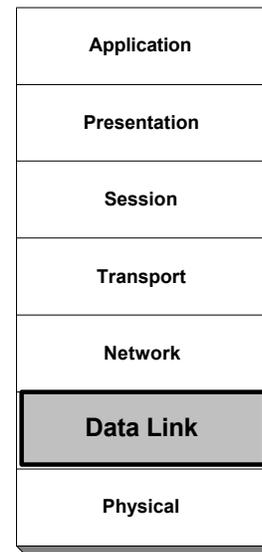


Figure B-26. 7 layer OSI Model

occurs. When a collision occurs, the stations involved wait a random period of time before attempting to transmit again, after first listening to make sure the network is available.

Ethernet is attractive to many organizations because it is designed to provide inexpensive, flexible, high speed, and maintainable Data Link services. However, because all stations on a Ethernet network share the available bandwidth⁷, as the network traffic increases, so do the number of network collisions – often resulting in lower overall throughput⁸.

The following strategies can be used to implement Ethernet networks.

- **Full-Duplex Ethernet.** For customers who require increased network bandwidth, but must leverage existing equipment, full-duplex Ethernet is a possible solution. Unlike Shared Ethernet, which uses hub⁹ technology, Full-Duplex Ethernet works with Ethernet switches, which allow stations to *simultaneously* transmit and receive data at 10Mbps – effectively doubling the bandwidth available with Shared Ethernet.
- **Switched Ethernet.** Using switch technology, Switched Ethernet provides the network entire bandwidth to each station for small, alternating intervals of time. Using this technique, station on the network does not share the bandwidth, as they do with Shared Ethernet. The Switch Ethernet technique can be implemented in conjunction with Full-Duplex Ethernet.

Although Ethernet hubs must be replaced with switching technology, organizations can typically continue to use existing Ethernet adapter cards and conduit infrastructures as they migrate from Shared or Full-Duplex Ethernet to Switched Ethernet. For this reason, Switched Ethernet is becoming a popular way to increase available network bandwidth, without significantly increasing costs. This growing popularity is one reason why Ethernet should remain the dominant desktop networking technology for years to come. This is illustrated in Figure B-27.

<i>LAN Technologies</i>	<i>Speed</i>	<i>Market share</i>
<i>Ethernet</i>	10Mbps - 100Mbps	75%
<i>Token Ring</i>	4Mbps or 16Mbps	15%
<i>Fast Ethernet</i>	100Mbps	2%
<i>FDDI</i>	100Mbps	<1%
<i>ATM</i>	25Mbps - 2.4Gbps	<1%

Figure B-27. LAN Technology: Speed and Market Share

Source: Alex Berson - Client/Server Architecture

- **Fast Ethernet.** With the arrival of economical, powerful desktop PCs and the subsequent distribution of complex applications, organizations have realized a need for increased network bandwidth. Providing data rates of up to 100Mbps, Fast Ethernet is being used to satisfy these needs. That is, leveraging the same data transmission formats (frames) and CSMA\CD techniques as the Ethernet variants that have already been described, Fast Ethernet provides 100Mbps data transmission rates. What is more, Fast Ethernet provides backward compatibility with slower Ethernet implementations (for example, 10Mbps Shared Ethernet).

Unlike migrating from Shared Ethernet to Switched Ethernet, upgrading from Ethernet to Fast Ethernet usually requires that network interface cards, hub/switch technology, and, in many cases, the conduit

⁷ Bandwidth describes the amount of data that can be transmitted on a physical medium at one time.

⁸ Throughput is the measure of data that is transmitted between to points. Throughput is a function of Bandwidth and data transmission rate.

⁹ Hubs are network devices that provide connectivity between stations by serving as a wiring nexus for the network.

infrastructure, be replaced. Nonetheless, Fast Ethernet is being installed in conjunction with Switched Ethernet, and is now commonly used for server-to-server connections, specialized applications with large data communication requirements, such as imaging, and even for desktop connectivity.

Token Ring is a Data Link protocol made popular by IBM that operates at data rates of either 4 Mbps or 16 Mbps and is often used in LANs. In Token Ring networks, stations are logically organized in a ring configuration and messages are passed from device to device in sequence via a data token, which confers the right to transmit data. That is, stations on a Token Ring network are allowed access to the network medium only when they are in possession on a token. Unlike Ethernet, which is based on CSMA\CD technology, network collisions are not an expected occurrence and, barring hardware errors, data transmissions are orderly and practically guaranteed to be successful on a Token Ring network. For these reasons, Token Ring networks are often referred to as “deterministic.”

As illustrated in Figure B-28, the “logical ring” configuration used by Token Ring is physically wired in a “star.” That is, each station is connected to the station directly preceding and following it on the network. This connection is facilitated via a multistation access unit (MAU), which is in turn connected to other MAUs via a ring. In this configuration, all stations are included within a logical ring and each station receives transmission only from the station preceding it and transmits messages only to the station following it. As a result of this configuration, stations must be capable of discovering their upstream and downstream neighbors. This requires Token Ring network interface cards to have a lot of error correction and connection circuitry that is not needed in Ethernet network. As a result, Token Ring equipment and networks are typically more expensive than Ethernet.

Like Ethernet, Full-Duplex and switched Token Ring has emerged. Full-Duplex Token Ring technology doubles the total bandwidth available to a station as explained in the Full-Duplex Ethernet section above. Using the switched technique, as explained earlier, stations do not simultaneously share the 16-Mbps network. Rather, the switch provides the entire network to each station for small, alternating intervals of time.

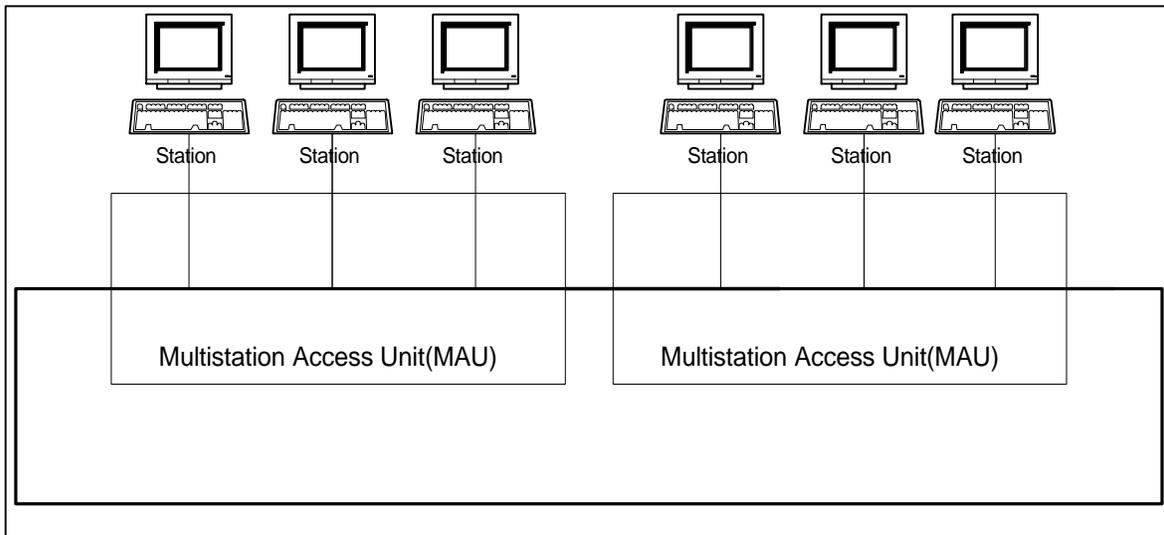


Figure B-28. Logical Token Ring configuration

Fiber Distributed Data Interface (FDDI), originated in 1982 by the American National Standards Institute (ANSI), is a standardized protocol for sending data over fiber optic cable at data rates up to 100Mbps. The FDDI protocol is based on token-passing and is implemented with a primary ring and a secondary ring. FDDI's redundant ring architecture, as illustrated in Figure B-29, provides fault tolerance

for the network. Under normal conditions only the primary ring is used. However, when a failure occurs on the primary ring, FDDI allows traffic to be routed via the secondary ring. In addition to increased fault tolerance and data rates, FDDI supports up to 1,000 stations on a single ring¹⁰, is not affected by electromagnetic interference, like wire-based technologies, and offers a high degree of security¹¹.

As illustrated in Figure B-29, FDDI is primarily used as network backbone technology. That is, many organizations use FDDI to provide capacity enhancements on congested LAN backbones, which interconnect enterprise-computing systems and large data-storage devices that conduct high-volume data transfer operations. However, with the emergence of low cost, high-speed networking alternatives, such as Fast Ethernet, FDDI use may decline in future years.

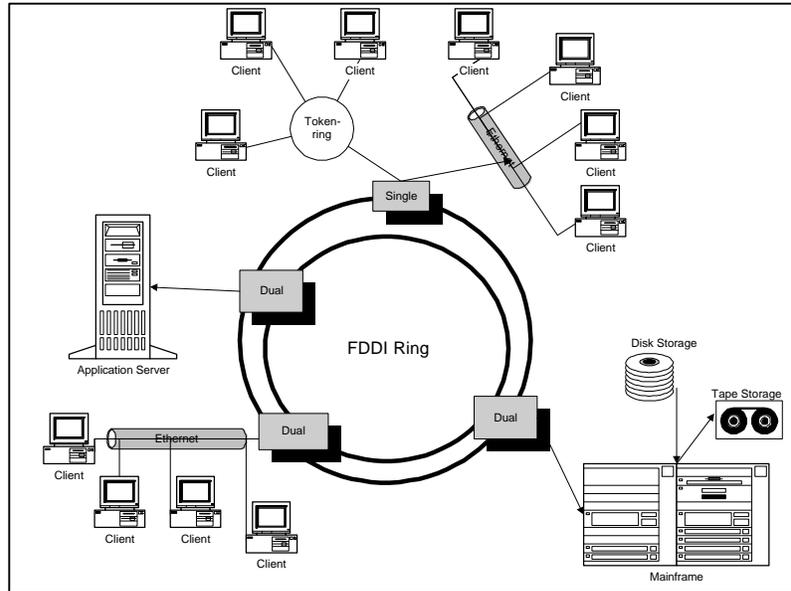


Figure B-29. FDDI's Redundant Ring Architecture

Asynchronous Transfer Mode (ATM) is a packet-switching¹² technology that uses small, fixed-length packets called cells that allow switching to be done entirely by hardware (which is much faster than software switching). ATM is designed to transfer data, with minimum delays, at very high speeds reaching up to 2Gbps. These characteristics make ATM ideal for a wide range of applications, including imaging, video, and multimedia. What is more, 25Mbps ATM can run over twisted-pair wiring, which is less expensive than the fiber-optic alternatives. However, ATM technology has several drawbacks, not least of which are equipment costs. ATM equipment is very expensive. For example, adapter cards can cost more than \$500.

It is widely assumed that ATM will be a significant networking technology in years to come. However, the absence of ratified standards has made integration complex and interoperability between heterogeneous vendor product offering difficult to attain. This has slowed ATM adoption, as has the requirement to replace almost every network-related system component (adapter cards, hubs, etc.) when transitioning to ATM. Nonetheless, because many technology alternatives are also expensive, ATM is being used increasingly in place of, for example, FDDI as a backbone networking technology.

¹⁰ Connecting stations to both the primary and secondary rings is optional and decreased the number of stations that may be connected to a single ring. 1,000 stations may be connected to a ring only when the primary ring is used.

¹¹ It is difficult to “tap” into the FDDI network without disrupting the network, and requires specialized equipment and skills to do so.

¹² Unlike circuit-switching networks, which define a static path for data to travel from one point to another, in packet-switching networks there is not direct connection between the sender and the receiver. Rather packet-switching networks use addressing information, which is transmitted with the data, to route transmission to their destination.

Frame Relay is a wide area connection protocol that uses switching technology to package data into frames (or packets) so they can be relayed from station to station. Designed to run over digital lines, frame relay is much simpler than other protocols, which operate on slower, noisy, error-prone analog lines. As a result, error-checking services are optional with Frame Relay.

Frame Relay operates on the principal that not every site with access to the network will need the network's bandwidth at the same time. That is, Frame Relay assumes that most of the time that a station is on the network it is not transmitting data. With this in mind Frame Relay interleaves data from a number of different sources, thus increasing the efficiency with which available bandwidth is used. Specifically, many technologies divide the available network into "virtual channels" and stations are assigned a channel, whether they were transmitting data or not. As a result, most of the time available bandwidth is not being fully used. As illustrated in Figure B-30, Frame Relay, uses a technique known as statistical multiplexing to allocate bandwidth only as needed. This result is more efficient bandwidth utilization.

Operating at 56Kbps to 2Mbps, Frame Relay is a good choice for applications such as LAN interconnections, where traffic volume may vary significantly, or where short bursts of very high bandwidth are required. (Some vendors are beginning to offer 45Mbps Frame Relay.)

Frame Relay is considered relatively inexpensive. Pricing varies from carrier to carrier and depends on specific parameters, such traffic volume, and committed (guaranteed) data rate, etc. However, aggressive pricing are contributing to the service's increasing popularity.

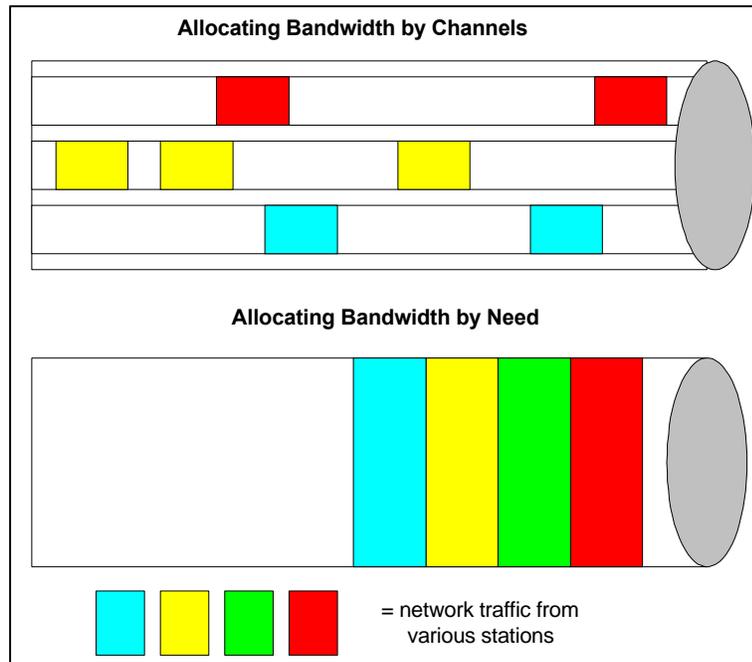


Figure B-30. Bandwidth Allocation: By Channels, and by Need

TECHNOLOGY: Online Transaction Processing Monitors

TECHNOLOGY DESCRIPTION:

Computer science described transactions in terms of four properties – know as the ACID properties. These properties are:

- **Atomicity** implies that a transaction is handled as a single, indivisible unit of operation; either all of the transaction's actions complete or none of the actions are committed (saved).
- **Consistency** ensures that relevant business rules are always enforced and that databases maintained by the system are always left in a consistent state. If the transaction cannot achieve a stable end-state, then the transaction must return (rollback) the system to its initial state. Consistency also implies that reality is reflected within the database. For example, if 100 students apply for financial aid, then the database should include application data for 100 students.
- **Isolation** indicates that a transaction cannot reveal its results to other concurrent transactions before commitment. That is, a transaction's behavior is not affected by other transactions that execute concurrently. Isolation assures that a transaction does not access or update data that is being updated by another transaction. This is particularly important in a high-volume system where hundreds of transactions execute more or less concurrently.
- **Durability** ensures that once a transaction is committed, the results are permanent and cannot be removed from the databases – except by another transaction. That is, system instabilities and failures in no way threaten the integrity of data. To ensure data durability, a transactional system provides mechanisms for backing up data and for logging (recording) transactional activities.

Simply put, a transaction can be defined as a sequence of database operations that transform the system from one consistent system state to a new consistent state. Transactions are characterized by those database operations involving the retrieval, insertion, update, and deletion of data.

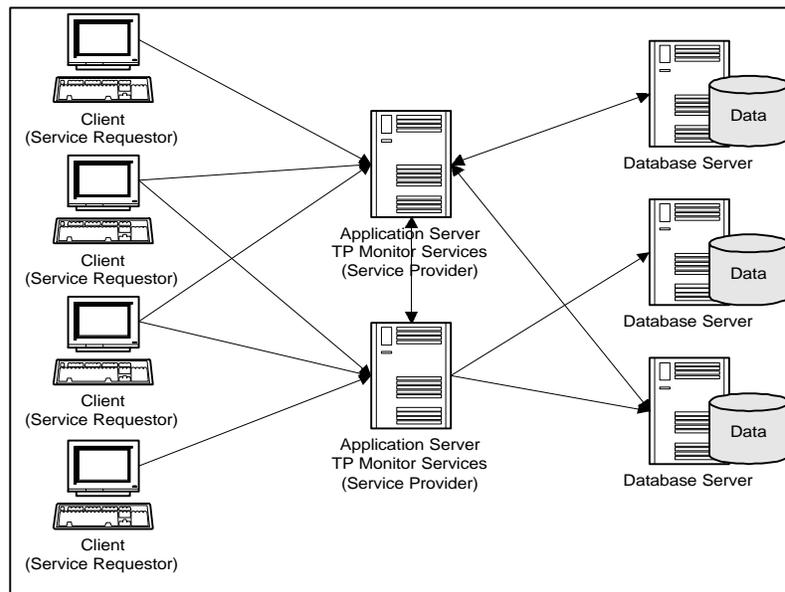


Figure B-31. TP Monitor Framework

Systems that specialize in managing the operations of many users as they retrieve, insert, update, and delete database records are commonly referred to as on-line transaction processing (OLTP) systems. OLTP systems have been in use for more than thirty years – within large mainframe and proprietary mid-range information systems. For example, IBM’s mainframe-based Customer Information Control System (CICS) is a very popular OLTP technology. More recently TP monitors – the software components that facilitate OLTP – have become available for the UNIX-based, open system environments commonly used within client-server and distributed systems.

Like mainframe-based TP monitors, today’s TP monitors are designed to provide centralized transaction management and control. However, unlike many of the more mature OLTP technologies, TP monitors are now being designed to work within the framework of modern distributed process and data strategies, as illustrated in Figure B-31. That is, TP monitors are now being used to 1) Facilitate interprocess communication between distributed application components; 2) Provide load balancing, priority scheduling, and process management services; and 3) Manage transactions that involve data stored within multiple, heterogeneous database management systems.

Having considered these OLTP-enabling services, many organizations are choosing to architect systems around TP monitors. That is, TP monitors are finding widespread acceptance with organizations that are not willing to exclusively rely on the OLTP services inherently provided by RDBMS technology or associated stored procedures (which can be used to extend the transactional capabilities of RDBMS technology). In particular, system architectures are relying on TP monitors to provide two mission-critical services:

- Process Management.
- Transaction Management.

The process management services provided by today’s distributed system TP monitors (monitors) allow monitors to do much more than just provide secure and reliable processing of online transactions. Monitors can be used to provide an extension to the operating system that adds substantial program-to-program communication, message-routing, and system configuration capabilities. Currently, most monitors support synchronous, as well as asynchronous communication. These communication models are illustrated in Figure B-32.

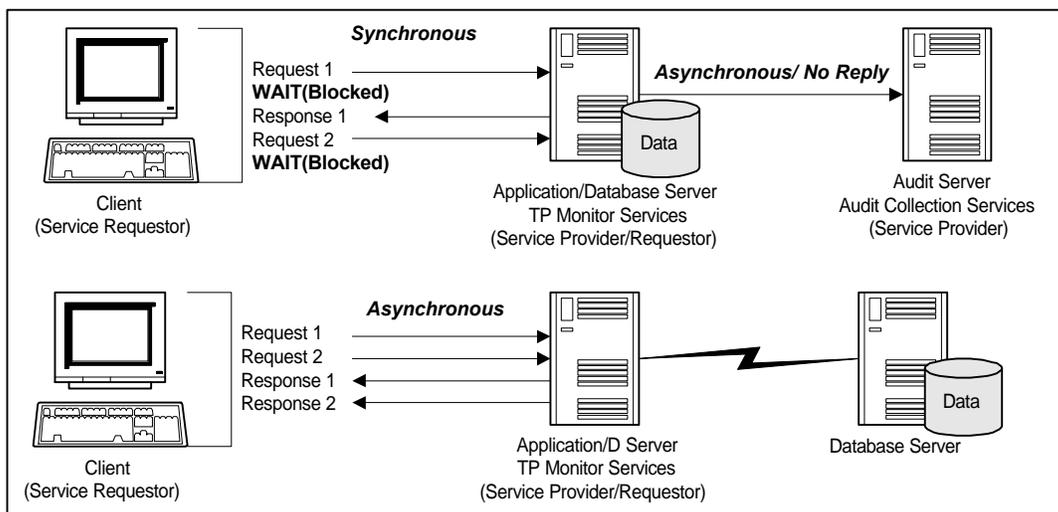


Figure B-32. Synchronous and Asynchronous Communication Model

Synchronous communication requires simultaneous availability of the client (service requesting) portion of the system and the server (service providing) portion of the system. Typically synchronous communication is “blocking.” That is the service requestor must wait for the service response before processing can be continued. For example, synchronous communication might be used to request and provide student information that is needed for subsequent student aid administration processing.

Unlike synchronous communication, asynchronous communication does not require the service provider to be available when a service request is issued. That is, application clients and servers do not have to be simultaneously available. Rather, the services may be requested and then serviced at a later date or time. Asynchronous communication is typically not blocking. That is, the service requestor is free to conduct further processing while previously issued service requests are being satisfied (responded to). For example, the EASI/ED target system might use asynchronous communication to request IRS validation of student income information. In this situation, the IRS batch cycles or other system workloads may not allow immediate request processing and response; however, because asynchronous and non-blocking communication is being used the EASI/ED target system is free to continue other operations while the IRS satisfies the request. Asynchronous and non-blocking communication might also be used in situations where the service provider is not required to respond to the client at all. For example, a client may request a service provider to capture performance or audit information. In this situation, asynchronous communication would be appropriate, as 1) a response from the service provider is not required and 2) the service provider’s ability to immediately satisfy requests should not affect client performance.

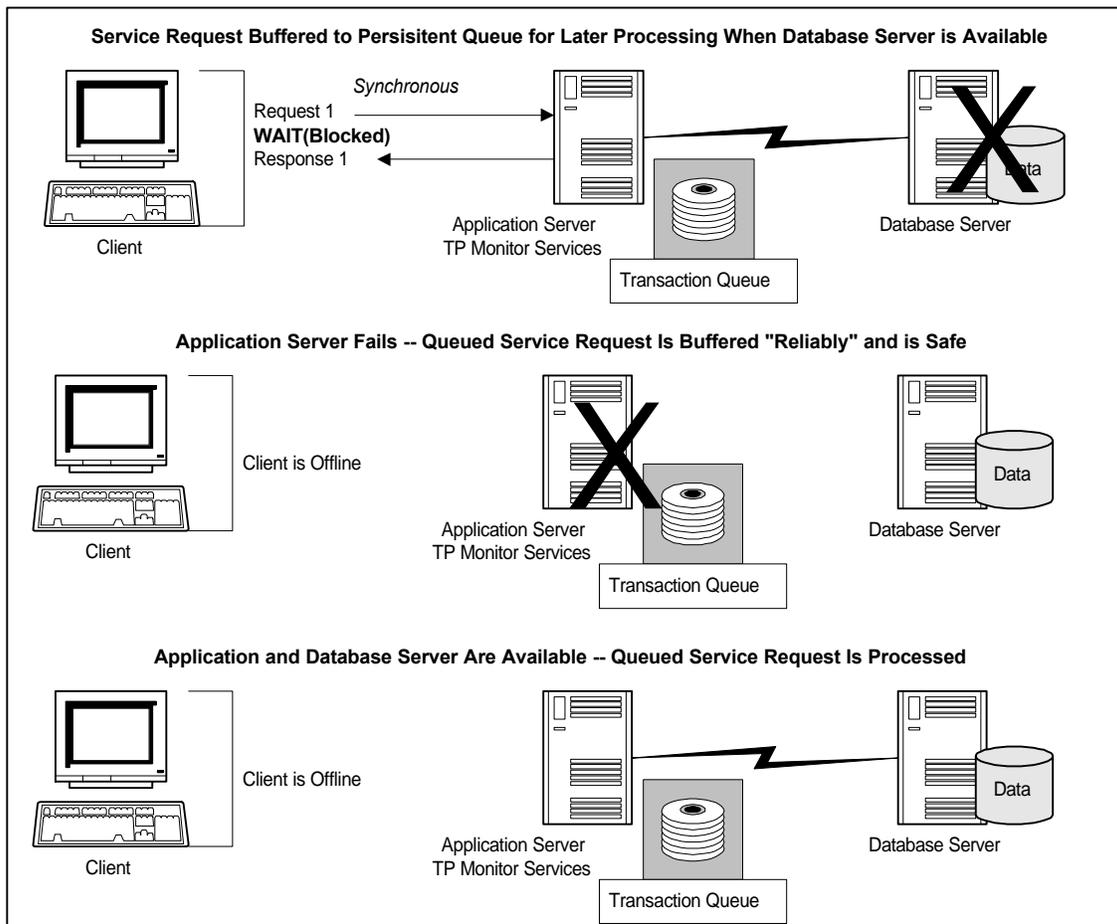


Figure B-33. Asynchronous Communication Model

Typically, TP monitors provide asynchronous communication services via message queues (as explained in the Middleware abstract) and many monitors even offer persistent or reliable queuing services. Persistent queuing services buffer transactions to disk to increase reliability should a connection fail. Persistent queuing introduces a great deal of application flexibility and can greatly increase “apparent” system availability. For example, using persistent queuing the EASI/ED system could accept student aid applications and buffer these applications to disk where they could be processed when the service provider was available. Using this configuration, students can submit applications, even when portions of the EASI/ED system are unavailable. What is more, queued transactions can be recovered and completed, even if a failure were to occur on the system on which the queue resides. This is illustrated in Figure B-33.

In addition to facilitating interprocess communication – à la middleware, TP monitors can also be used to manage distributed system communication processes. Consider the process-per-user memory management strategy described within the RDBMS abstract. This memory management strategy provides a separate process and address space for each client connection to the database, whether the connection is actively used or not¹³. As illustrated in Figure B-34, in system architectures where the RDBMS is directly accessed by clients (service requestors), if there are 10,000 clients then 10,000 client processes would have to be instantiated and each of these 10,000 processes would consume system resources. This configuration requires substantial system resources, as the database server must not only support database I/O operations, but the operating system must also manage the considerable workload generated by 10,000 client connections.

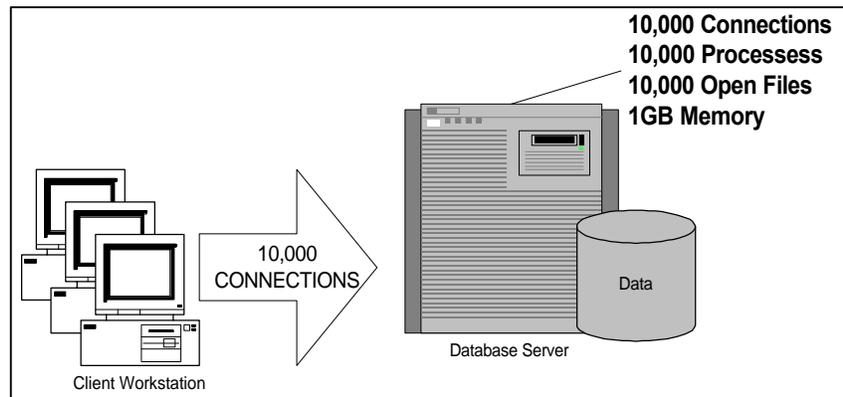


Figure B-34. Architecture without a TP Monitor

To resolve these architectural deficiencies, TP monitors can be used to dynamically assign memory – thereby:

- Increasing the number of active users that can address a system.
- Reducing the system’s hardware requirements.
- Boosting system response time.

Specifically, the TP Monitor removes the process-per-client requirement by “funneling” incoming client requests to shared server processes. TP monitor “funneling” is based on the premise that process-per-client connections are under utilized and that resources associated with these connections can be shared by many clients. However, it is important to note that clients do not concurrently share process address space. Rather they use and then “release” these resources for use by other clients. As a result:

- Clients receive the enhanced fault tolerance of the process-per-client strategy.

¹³ The process-per-user configuration effectively isolates client operation and increases fault tolerance. However, this strategy also consumes a considerable amount of system resources and yields comparatively weak performance.

- Concurrent database server connections are reduced – thus reducing database server resource requirements.

What is more, many TP monitor technologies enhance these process management techniques with load balancing services, which dynamically start new processes when the number of incoming client requests exceed the number of available shared server processes. TP monitor process “funneling” is illustrated in Figure B-35.

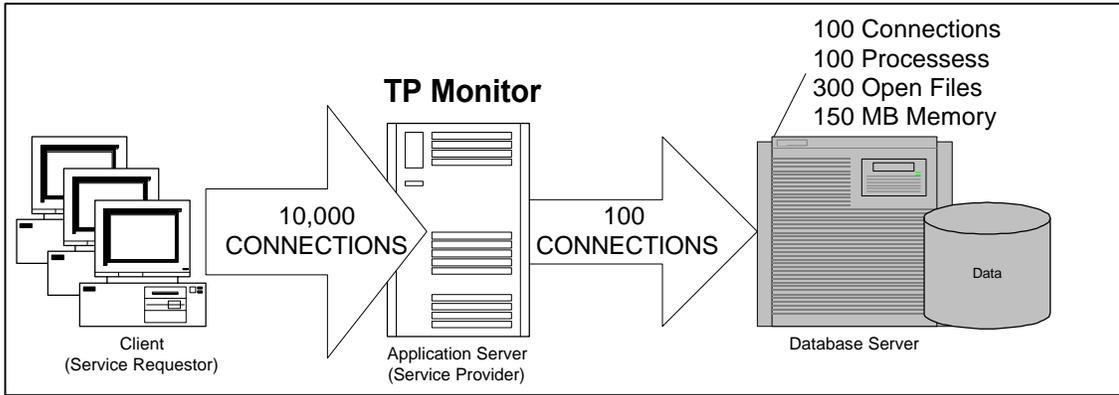


Figure B-35. Architecture with a TP Monitor

Although TP monitors provide valuable process management services, as described in the previous paragraphs, their forte is distributed transaction management. That is, using the two-phase commit protocol, TP monitors provide the capability to manage distributed flat transactions involving multiple, heterogeneous database management systems. Figure B-36 illustrates distributed transaction management.

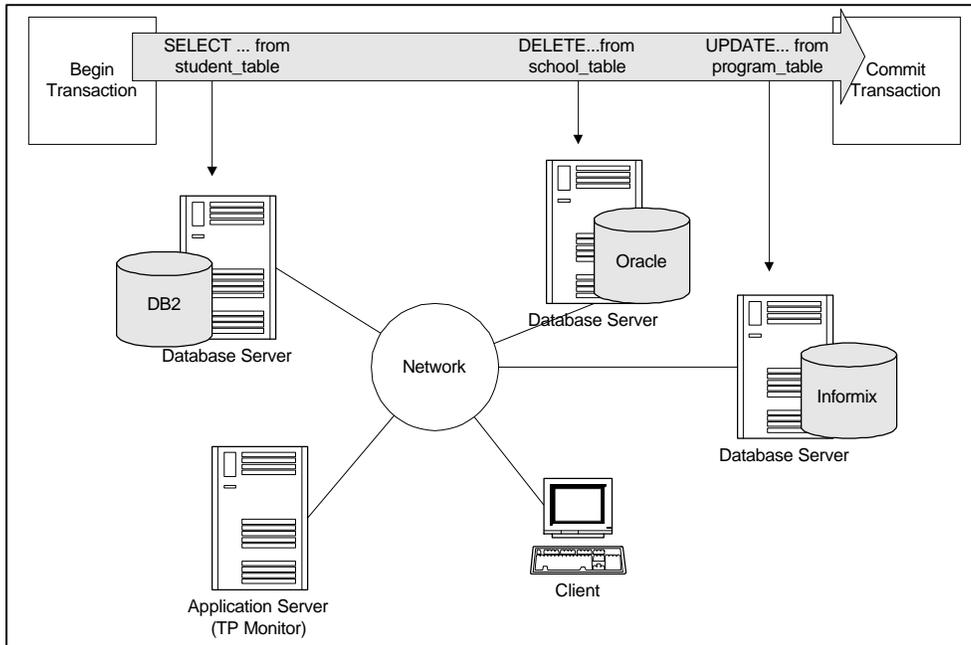


Figure B-36. Distributed Transaction Management

The term “flat” transaction indicates that all operations defined within the bounds of the transaction must be successfully executed in order for any part of the transaction to be committed (saved). Simply put, flat transactions are an all-or-nothing proposition – operations are atomic and cannot be partially saved or rolled back. Distributed flat transactions that involve multiple database systems require a high degree of system parallelism. That is, the various systems, which host the databases involved in the transaction, must be simultaneously available.

In a distributed environment, where a transaction seeks to modify multiple databases residing on two or more geographically dispersed systems, maintaining transaction integrity can become complicated. For this reason, TP monitors commonly use the two-phase commit protocol to guarantee the ACID properties of all executed flat transactions.

The two-phase commit guarantees the integrity of distributed data by ensuring that transaction operations are either finalized in all of the separate databases, or are fully backed out of each of the databases. That is, when a transaction signals that it has finished processing, the transaction coordinator (the TP monitor) notifies each transaction participant (database management systems) to “prepare” for transaction completion. Once all participants have responded to the “prepare” phase, the coordinator determines whether to commit or rollback all database operations defined within the transaction. That is, if all transaction participants respond affirmatively to the “prepare” phase, then all transactions are committed to complete the second phase of the transaction. However, if any participant refuses to prepare for transaction commitment or does not respond to the “prepare” broadcast, then the coordinator informs all participants to rollback the transaction.

COMMERCIAL OFFERINGS:

Vendor	Product
BEA Inc. /Novell Inc.	Tuxedo
Transarc Inc. /IBM Corp.	Encina
IBM Corp.	CICS
Microsoft Corp.	Transaction Server
AT&T	Top End

TECHNOLOGY: Web-based Distributed Systems

TECHNOLOGY DESCRIPTION:

As illustrated in Figure B-37, in recent years distributed systems have evolved from “fat client” Remote Data Access architectures, where only the DBMS services were distributed, to Remote Presentation architectures, where application and data management logic are distributed to specialized hardware environments. Today these architectures continue to evolve – largely because of the proliferation of the Internet, the World Wide Web, and related technologies.

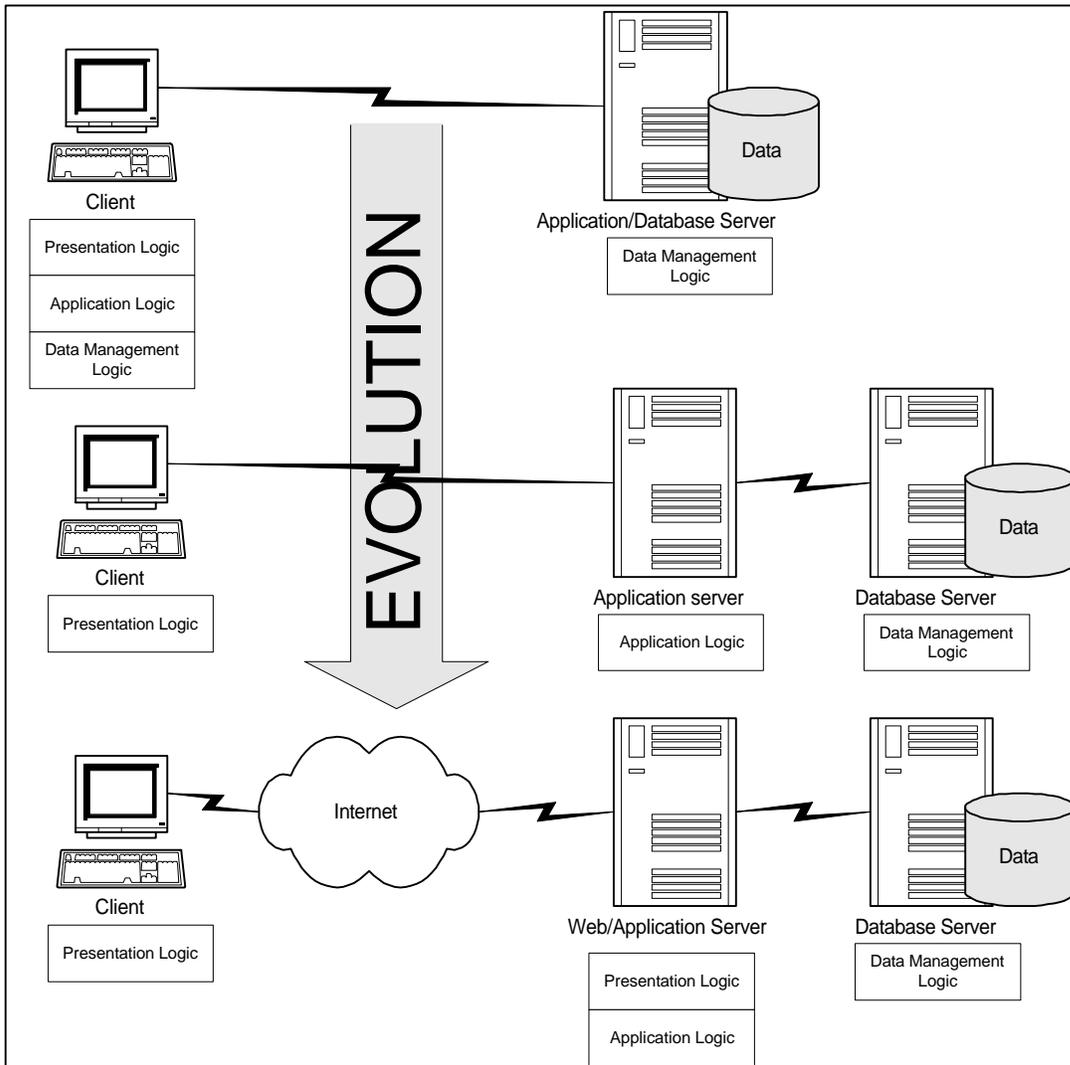


Figure B-37. Evolution of Distributed Systems

Web-based distributed system architectures are finding widespread acceptance with organizations for a variety of reasons, not least of which is the promise of reduced communication costs that may result as organizations leverage public networks – the Internet – to create instantaneous global communication channels with users, customers, suppliers, etc. Other reasons that these architectures are being adopted include:

- **Application Platform Independence.** Within Web-based distributed system architectures, Web browsers are used, almost exclusively, to provide client application presentation services. Today, Web browsers are available for nearly any platform, including Windows, Macintosh, Windows NT, and UNIX. What is more, Web browsers are inexpensive, if not freely distributed, and are readily available. For example, many PC operating systems bundle a Web browser with the operating system – a la Windows95 or NT.

To a varying degree, commercially and freely distributed Web browsers provide support for the HyperText Mark-up Language (HTML), which includes a standardized set of conventions for organizing and displaying data. That is, Web browsers do not leverage presentation logic embedded within proprietary operating system environments. As a result, Internet-based distributed systems provide consistent presentation services that many users are already familiar with, regardless of the user’s operating environment.

- **Simplified Configuration Management and Control.** The client platforms within many Web-based architectures are very “thin” – hosting only a Web browser and no custom code. As a result, system modifications and configuration changes are only made to system components residing on Web, application, and data management servers. Likewise, most new system “features” need only be deployed to a very small number of machines – the servers – where they can be immediately accessed by system users. As a result, all clients access applications via the system’s Web server and there is no need to worry about which clients are using which version of the application.

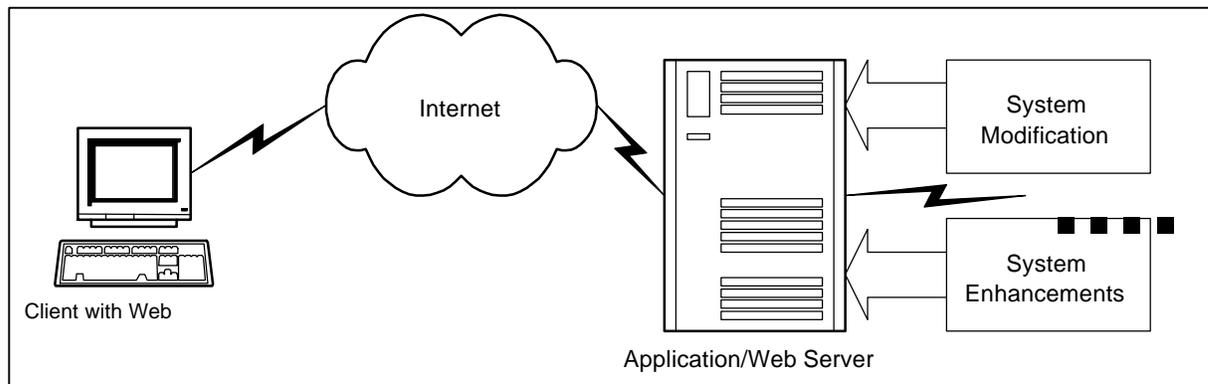


Figure B-38. Web-Based Client/Server Architecture

Despite these advantages, Web-based distributed system architectures can be complex and technically challenging. Specifically, many Internet and World Wide Web technologies are immature, untested, and unfamiliar to many system integrators and architects. As a result, the project performance and level-of-effort estimates provided by many developers may be inaccurate or at the very least suspect. Other complications associated with Internet-based system architectures include:

- **Limited System Control.** Within more traditional distributed system architectures the network infrastructure that interconnects system components, such as clients and servers, is controlled and administered by the organization. This is not the case when public network resources are used as the information delivery medium. That is, the organization does not have the authority to administer, maintain, or modify public network resources. As a result, system performance, which may be significantly affected by the network, is largely dependent upon an uncontrolled and in many ways unpredictable resource – the Internet.
- **Complex Security.** Organizations typically “expose” their systems to the Internet in order to leverage the Internet’s ability to create instantaneous global communication channels for sharing information. However, this exposure also introduces complexity and, in many cases, the requirement to implement complex multilevel security solutions (MSS). These solutions allow systems to contain information

with different sensitivities, while simultaneously preventing users from obtaining access to information for which they lack authorization. In short, exposed systems must be capable of convenient:

- **Identification and Authentication** – the ability to verify a user’s identity and a message’s authenticity.
- **Access Control and Authorization** – the protection of information from unauthorized access.
- **Confidentiality** – the protection of information from unauthorized disclosure.
- **Integrity** – the protection of information from unauthorized modification or accidental loss.
- **Nonrepudiation** – the ability to prevent users from denying they have sent or received information.

Additionally, exposed systems must provide mechanisms for securing information communicated via uncontrolled and widely accessible public networks. These security mechanisms must be implemented via “open” technologies, which are supported by the wide variety of operating environments and commercially/publicly available Web browsers in use today.

Despite the challenges associated with Web-based distributed systems, the promise and benefits of the Internet paradigm continue to attract organizations. With this in mind, Figure B-39 illustrates the technologies that provide the underpinnings of the Web. These technologies include the Web browser, HyperText Markup Language (HTML), HyperText Transfer Protocol (HTTP), TCP protocol, and the Web server.

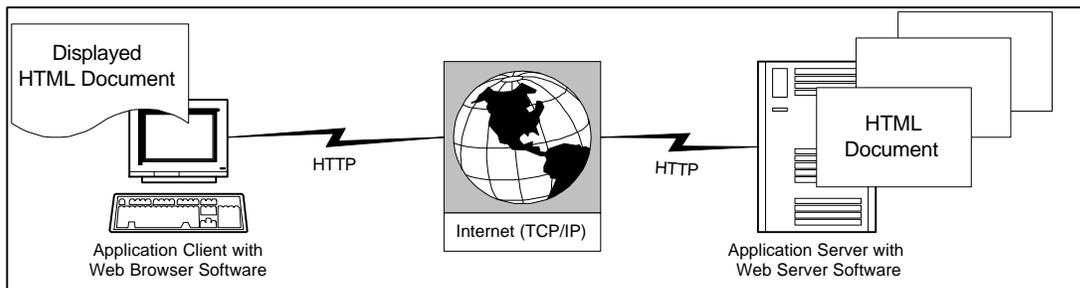


Figure B-39. World Wide Web Base Technologies

Web documents – which compose the user’s graphical user interface – are ASCII text files that include HTML commands and user information (“content”). HTML commands (“tags”) describe how the file’s content should be organized and displayed by the Web browser (browser) and provide references (“HyperLinks”) to other documents.

Web documents are stored on and retrieved from Web servers. That is, browsers are used to locate, specify, request, interpret, and display Web documents maintained on Web servers located throughout the Internet. Specifically, via a browser, users request information by specifying the Universal Resource Locator (URL) of required Web documents.

A URL is a general-purpose naming scheme that is used to locate and request Internet resources, such as Web documents. As illustrated in Figure B-40, a URL typically has four parts: protocol, server, port, and target.

- **Protocol** – identifies which Internet protocol should be used to access resources. Examples include file transfer protocol (FTP), News, Mailto, Gopher, and HTTP.

- **Server** – identifies the Internet host domain name or Internet Protocol (IP) address of the server on which the desired resource resides.
- **Port** – identifies the program running on the server that will respond to the user’s resource request – the Web server software.
- **Target** – identifies the location of the resource on the server. For example, the target might be the directory path and file name of an HTML document.

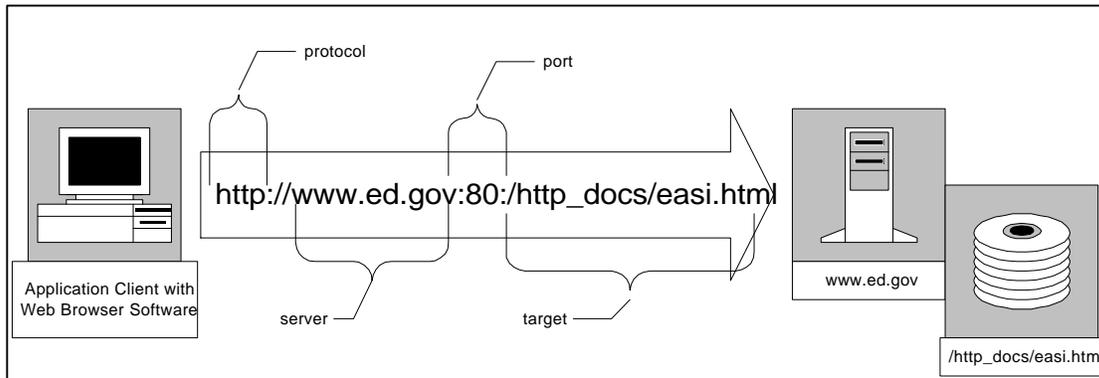


Figure B-40. Universal Resource Locator (URL)

Once the Web document request has been specified via a URL, it is communicated to the Web server identified within the URL. This communication is facilitated via HTTP. Much like the remote procedure call (RPC) described in the Middleware abstract, the HTTP protocol is synchronous¹⁴ and 1) establishes a client/server connection, 2) transmits browser requests and server responses, and 3) terminates the client/server connection.

Once contacted via the HTTP protocol, the Web server processes the users request and returns the requested HTML document. Upon receipt of the requested document, the browser interprets the HTML commands contained therein and displays the contents for the user.

HTML documents may include tags that simply specify font types and colors, hypertext links, or embedded graphics. However, as illustrated in Figure B-41, HTML can also be used to develop forms, which include text fields, radio boxes, tables, combo boxes, check boxes, buttons, and other “widgets” that are commonly used within event-driven graphical user interface (GUI) applications. Working in conjunction with Common Gateway Interface (CGI) or similar server-based application components, HTML forms allow users to extend the Web paradigm. That is, with the introduction of forms and CGI, the Web became more than just a document publication tool – it became an enabling technology for globally distributed information management systems.

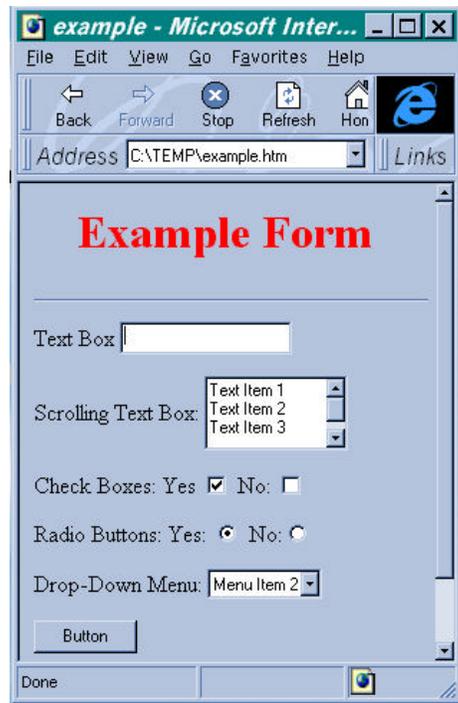


Figure B-41. Example of an HTML Form

¹⁴ Synchronous protocols require the simultaneous availability of the client and server.

CGI applications are typically UNIX “filters” and are often developed using operating system shell facilities, familiar scripting utilities, such as Perl, Awk, and Sed, or compiled programs – typically written using the C or C++ language.

Web servers invoke CGI applications, which in turn usually interact with a back-end application component, such as a RDBMS or an online transaction processing monitor (TP monitor). That is, the Web server parses the incoming data from the application client browser. If the data received from the browser requests CGI services, the Web server instantiate environmental variables for communicating with the CGI application and then invokes the requested CGI application. Once invoked, the CGI application reads the environmental variables created by the Web server and receives the form contents, which the Web server writes to standard output. Having received the form contents, the CGI application processes the contents as designed, formats results in HTML, and sends the HTML results to the Web server via standard output. As illustrated in Figure B-42, once the Web server receives the CGI application results, they are forwarded to the browser via the HTTP protocol.

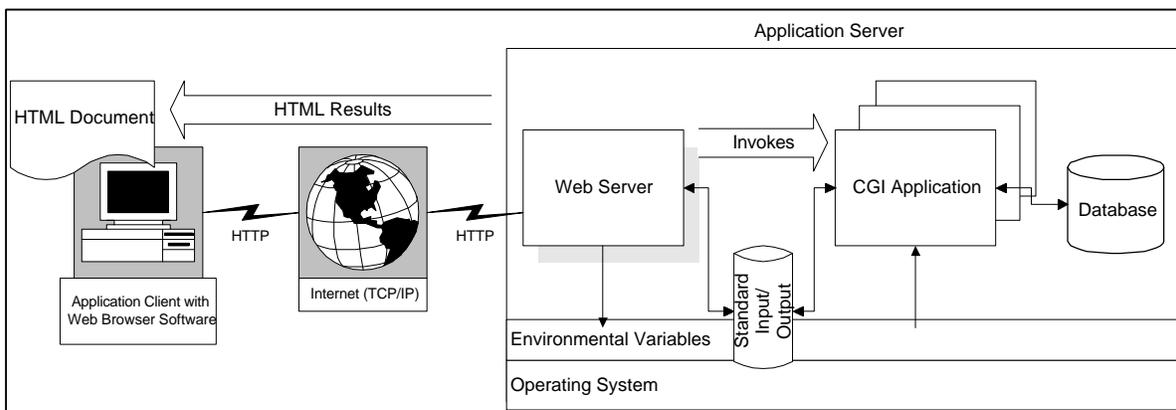


Figure B-42. CGI Application Execution

CGI makes it possible for globally distributed clients to access and use the services provided by a variety of back-end services, such as RDBMSs, TP monitors, middleware, workflow, file systems, data warehouses, etc. Having realized the potential of this technology, many software vendors now offer product integration solutions (“gateways”) that accept CGI requests – many even provide packaged solutions that dynamically transform, for example database query results, into the HTML format required by Web servers and browsers. However, CGI-solutions can introduce performance constraints and security concerns. (Security is described within the Distributed Security Abstract.)

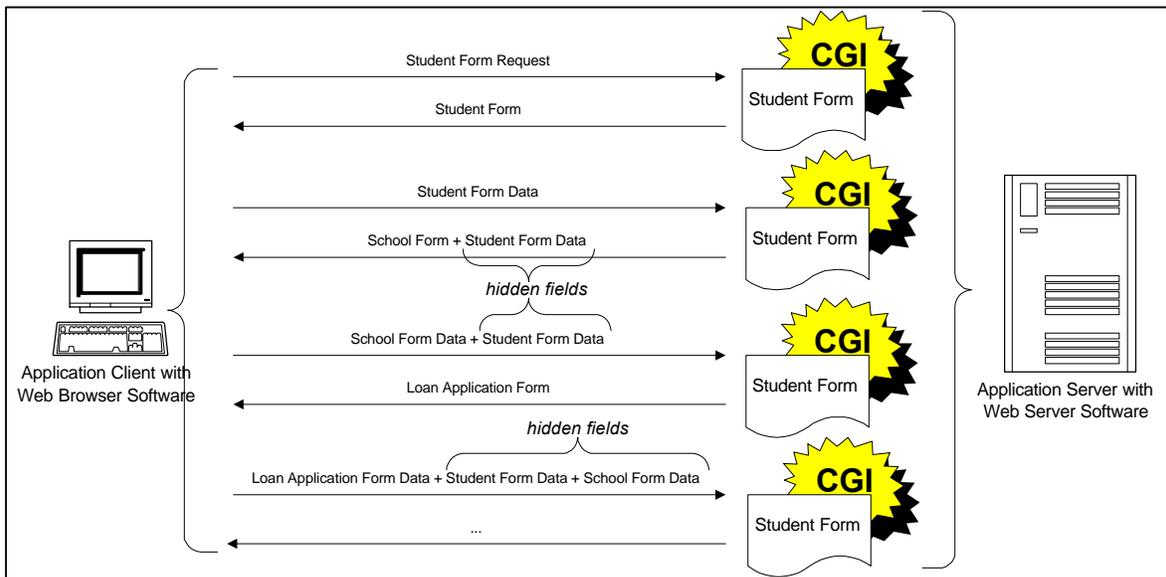


Figure B-43. CGI Uses Hidden Fields to Maintain Data from One Form Within a Second

CGI performance constraints are largely introduced by the stateless nature of HTTP and CGI. As described earlier, HTTP is a synchronous protocol that 1) establishes a client/server connection, 2) transmits browser requests and server responses, and 3) terminates the client/server connection. As a result, multiple client/server connections are required for complex transactions that involve the request and submission of multiple forms. However, because HTTP and CGI are stateless, these client/server connections and the information communicated therein is not technically recognized as a single transaction. To solve this problem CGI uses hidden fields to “associate” information requested and submitted via multiple forms. For example, if a business transaction involved two forms – one to capture student information and a second to capture the student’s school information – CGI would use hidden fields to store collected student data (from the first form) within the second form as illustrated in Figure B-43. Obviously this is an inefficient use of networking resources, but is required because the CGI application is not capable of maintaining the student data within system memory. That is, CGI applications are themselves stateless – terminating with each client/server connection.

In addition to efficiencies associated with the stateless nature of HTTP and CGI, performance deficiencies can also result from Web-based distributed architectures that relying solely on server-based logic, like CGI, and the Web’s request-response paradigm. That is, without the availability of client-based edit and validation services, for example, the accuracy and completeness of data can not be determined without first submitting the data to the Web Server and associated application components. This makes for inefficient use of network and server resources. One solution to this architectural challenge is Java.

Java is an object-oriented programming language that allows developers to write scripts (JavaScript) or small programs – applets – which can be downloaded to and executed by Java-compatible browsers (Java can also be used to implement server-side application components). In short, Java allows executable code to be distributed across the Web along with HTML data. In doing so, Java allows data processing to occur before documents are submitted to the Web server. What is more, as illustrated in Figure B-44, Java is managed from the server side of the distributed systems architecture. As a result, the benefits of simplified configuration management and control are not sacrificed when Java is used within Web-based distributed system architectures.

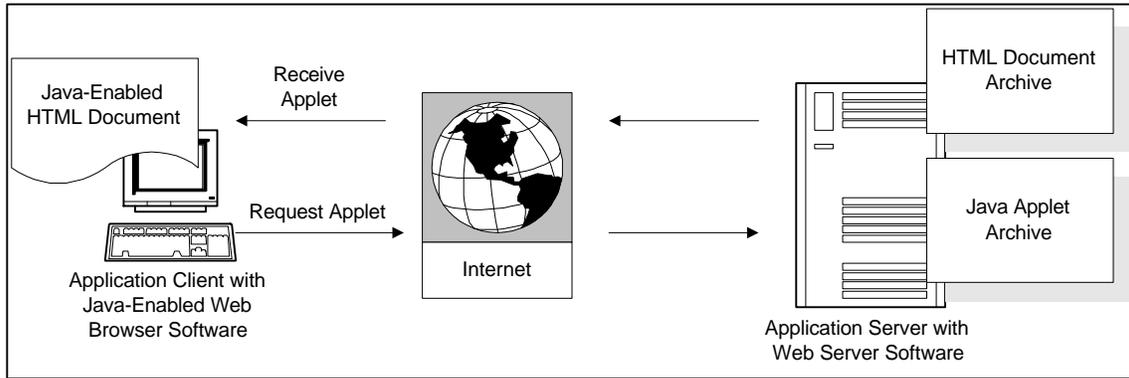


Figure B-44. Java is Managed from the Server Side of the Distributed Systems Architecture

COMMERCIAL OFFERINGS:

Vendor	Product
Microsoft Corp.	Internet Information Server, Internet Explorer, Internet Studio
Netscape Inc.	Netscape Navigator/ Navigator Gold, LiveWire Pro, SuiteSpot, Enterprise Server, Catalog Server, Fast Server, News Server, Proxy Server,
IBM Corp.	Lotus Notes, NetCommerce Server, Visual Age
Oracle Corp.	WebSystem: WebServer, PowerBrowser, CommerceServer
Sun Microsystems Inc.	HotJava