4

Client/Server and Databases

4.1 INTRODUCTION

Storing Data and the Database

Server translates the ink/paper storage model into an electronic/magnetic media storage model, but the fundamental arrangement is the same. The basic building block (his computer equivalent of information-on-paper) is called data. Data is information in its simplest form, meaningless until related together in some fashion so as to become meaningful. Related data is stored on server's disk under a unique name, called file. Related file are gathered together into directories, and related directories are gathered together into larger and larger directories until all the required information is stored in a hierarchy of directories on the server's hard disk.

The server's "filing cabinet" is a database; it offers a number of advantages over the paper model. A particular file can be searched electronically, even if only remembering a tiny portion of the file contains.

A database, generally defined, is a flexible, hierarchical structure for storing raw data, which facilitates its organization into useful information. All data on computer is stored in one kind of database or another. A spreadsheet is a database, storing data in an arrangement of characters and formatting instructions. What a database does, then, is breakdown information into its most fundamental components and then create meaningful relationships between those components. We depend on databases of varying configurations and complexity for all our computerized information need.

The Fig. 4.1 illustrates the evolution of database technology from the first computer in late 1950's to the object-oriented database technologies.

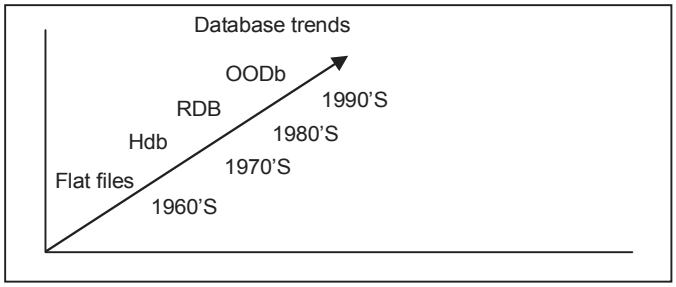

Fig.4.1: Evolution of Database Technologies

Using a database you can tag data, relating it to other data in several different ways, without having to replicate the data in different physical locations. This ability to access and organize data in a flexible manner without making physical copies of it (and thus preserving the integrity of the information at its most basic level) is what has lead to the increasing use of client/server technology as a widespread business information model.

Database System Architectures

Before proceeding to understand the Client/Server database it is quite essentials to have a brief introduction about the other available architecture of database systems.

Client/Server database system: The functionality is spilted between a server and multiple client systems, i.e., networking of computers allows some task to be executed on server system and some task to be executed on client system.

Distributed database system: Geographically or administratively distributed data spreads across multiple database systems.

Parallel database system: Parallel processing within computer system allows database system activities to be speeded up, allowing faster response to transaction; queries can be preceded in a way that exploits the parallelism offered by the underlying computer system.

Centralized database system: Centralized database systems are those run on a single system and do not interact with other computer systems. They are single user database systems (on a PC) and high performance database system (on high end server system).

4.2 CLIENT/SERVER IN RESPECT OF DATABASES

4.2.1 Client/Server Databases

Servers exist primarily to manage databases of information in various formats. Without the database, servers would be impractical as business tools. True, you could still use them to share resources and facilitate communication; but, in the absence of business database, a peer-to-peer network would be a more cost effective tool to handle these jobs. So the question of client server becomes a question of whether or not your business needs centralized database. Sharing and communications are built on top of that.

A Database Management System (DBMS) lies at the center of most Client/Server systems in use today. To function properly, the Client/Server DBMS must be able to:

- Provide transparent data access to multiple and heterogeneous clients, regardless of the hardware, software, and network platform used by the client application.
- Allow client request to the database server (using SQL requests) over the network.
- Process client data requests at the local server.
- Send only the SQL result to the clients over the network.

A Client/Server DBMS reduces network traffic because only the rows that match the query are returned. Therefore, the client computer resources are available to perform other system chores such as the management of the graphical user interface. Client/Server DBMS differ from the other DBMSs in term of where the processing take place and what data are sent over the network to the client computer. However, Client/Server DBMSs do not necessarily require distributed data.

Client/Server systems changes the way in which we approach data processing. Data may be stored in one site or in multiple sites. When the data are stored in multiple sites, Client/Server databases are closely related to distributed databases.

4.2.2 Client/Server Database Computing

Client/Server database computing evolved in response to the drawbacks of the mainframe (very expensive operating cost because they require specialized operational facilities demand expensive support, and do not use common computer components), and PC/file server computing environments (the drawback of PC-based computing is that all RDBMS processing is done on the local PC, when a query is made to the file server, the file server does not process the query, instead, it returns the data required to process the query, this can result in decreased performance and increased network bottlenecks). By combining the processing power of the mainframe and the flexibility and price of the PC, Client/Server database computing encompasses the best of both words.

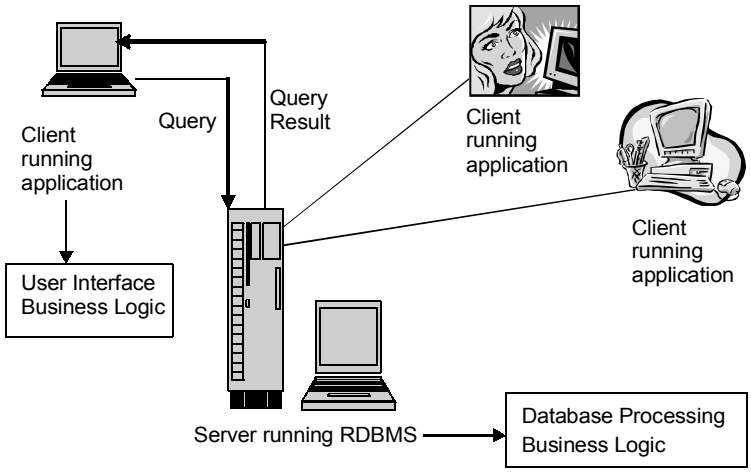

Fig. 4.2: Client/Server Database Computing

Client/Server database computing can be defined as the logical partition of the user interface, database management, and business; logic between the client computer and server computer. The network links each of these processes. The client computer, also called workstation, controls the user interface. The client is where text and images are displayed to the user and where the user inputs data. The user interface can be text based or graphical based. The server computer controls database management. The server is where data is stored, manipulated, and stored. In the Client/Server database environment, all database processing occurs on the server.

Business logic can be located on the server, on the client, or mixed between the two. This type of logic governs the processing of the application.

Client/Server database computing vs. Mainframe and PC/file server computing

Client/Server database computing is preferred in comparison to other database computing. Following are the reasons for its popularity:

Affordability: *Client/Server computing can be less expensive than mainframe computing. The underlying reason is simple: Client/Server computing is based on an open architecture, which allows more vendors to produce competing products, which drives the cost down. This is unlike mainframe-based systems, which typically use proprietary components available only through a single vendor. Also, Client/Server workstations and servers are often PC based. PC prices have fallen dramatically over the years, which has led to reduce Client/Server computing costs.*

Speed: *The separation of processing between the client and the server reduces the network bottlenecks, and allows a Client/Server database system to deliver mainframe performance while exceeding PC/file server performance.*

Adaptability: *The Client/Server database computing architecture is more open than the proprietary mainframe architecture. Therefore, it is possible to build an application by selecting an RDBMS from one vendor, hardware from another vendor. Customers can select components that best fit their needs.*

Simplified data access: *Client/Server database computing makes data available to the masses. Mainframe computing was notorious for tracking huge amounts of data that could be accessed only by developers. With Client/Server database computing, data access is not limited to those who understand procedural programming languages (which are difficult to learn and require specialized data access knowledge). Instead, data access is providing by common software products tools that hide the complexities of data access. Word processing, spreadsheet, and reporting software are just a few of the common packages that provide simplified access to Client/Server data.*

4.3 CLIENT/SERVER DATABASE ARCHITECTURE

Relational database are mostly used by Client/Server application, where the server is a database server. Interaction between client and server is in the form of transaction in which client makes database request and receives a database response.

In the architecture of such a system, server is responsible for maintaining the database, for that purpose a complex database management system software module is required.

Various types of applications that make use of the database can install on client machine. The "glue" that ties client and server together is software that enables the client to make request for access to the server's database, that is SQL (Structured Query Language), shown in the Fig. 4.3 given below:

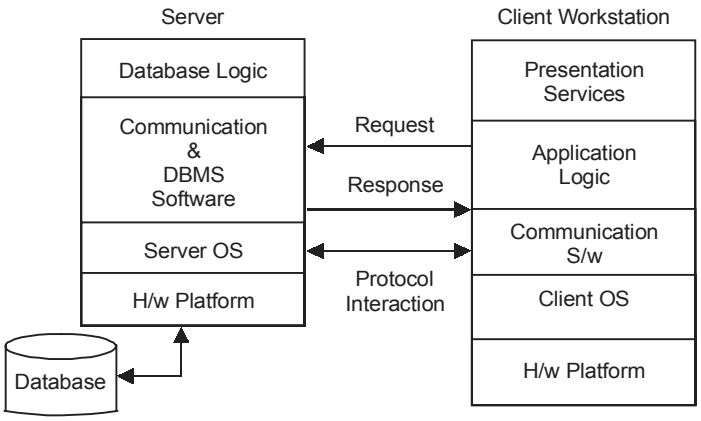

Fig.4.3: Client/Server Database Architecture

According to this architecture, all the application logic (software used for data analysis) is residing on the client side, while the server is concerned with managing the database. Importance of such architecture depends on the nature of application, where it is going to be implemented. And the main purpose is to provide on line access for record keeping. Suppose a database with million of records residing on the server, server is maintaining it. Some user wants to fetch a query that result few records only. Then it can be achieved by number of search criteria. An initial client query may yield a server response that satisfies the search criteria. The user then adds additional qualifiers and issues a new query. Returned records are once again filtered. Finally, client composes next request with additional qualifiers. The resulting search criteria yield desired match, and the record is returned to the client. Such Client/Server architecture is well-suited for such types of applications due to:

- Searching and sorting of large databases are a massive job; it requires large disk space, high speed CPU along with high speed Input/Output architecture. On the other hand, in case of single user workstations such a storage space and high power is not required and also it will be costlier.
- Tremendous traffic burden is placed on the network in order to move the million of records to the clients for searching, then it is not enough for the server to just be able to retrieve records on behalf of a client; the server needs to have database logic that enables it to perform searches on behalf of a client.

Various types of available Client/Server Database Architecture are discussed in detail; in the section given:

(*i*) Process-per-client architecture.

(*ii*) Multi-threaded architecture.

(*iii*) Hybrid architecture.

(*i*) Process-per-client architecture: As the name reveals itself server process considers each client as a separate process and provides separate address space for each user. Each process can be assigned to a separate CPU on a SMP machine, or can assign processes to a pool of available CPUs. As a result, consumes more memory and CPU resources than other schemes and slower because of process context switches and IPC overhead but the use of a TP Monitor can overcome these disadvantages. Performance of Process-per-client architecture is very poorly when large numbers of users are connecting to a database server. But the architecture provides the best protection of databases.

Examples of such architecture is DB2, Informix, and Oracle6, Fig. 4.4 illustrates such architecture.

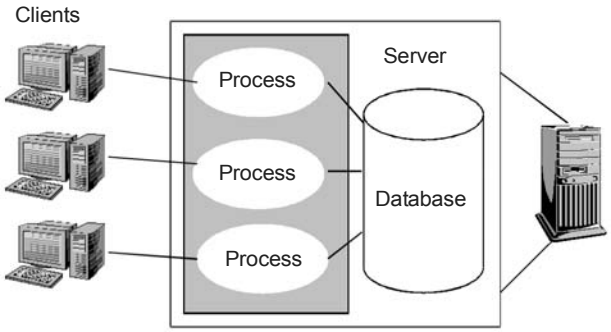

Fig.4.4: Process-per-client Architecture

(*ii*) Multi-threaded architecture: Architecture supports a large numbers of clients running a short transaction on the server database. Provide the best performance by running all user requests in a single address space. But do not perform well with large queries. Multi-threaded architecture conserves Memory and CPU cycles by avoiding frequent context switches. There are more chances of portability across the platforms.

But it suffers by some drawback first in case of any misbehaved user request can bring down the entire process, affecting all users and their requests and second long-duration tasks of user can hog resources, causing delays for other users. And the architecture is not as good at protection point of view. Some of the examples of such architecture are: Sybase, Microsoft SQL Server, and illustrated in Fig. 4.5.

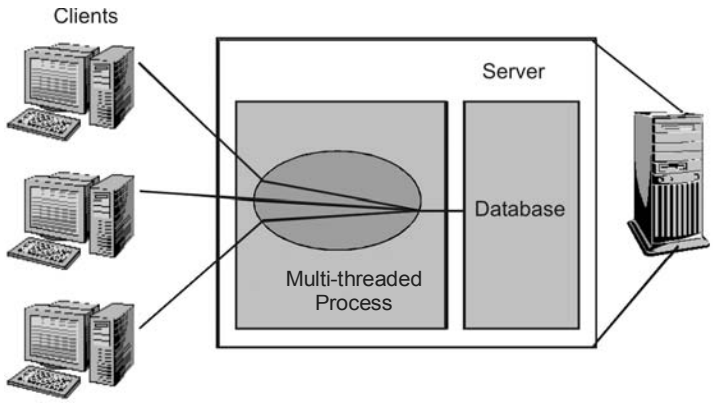

Fig.4.5: Multi-threaded Architecture

(*iii*) Hybrid architecture: Hybrid architecture provides a protected environment for running user requests without assigning a permanent process for each user. Also provides the best balance between server and clients. Hybrid architecture of Client/Server Database is basically comprised of three components:

- Multi-threaded network listener: Main task of this is to assign client connection to a dispatcher.
- Dispatcher processes: These processes are responsible for placing the messages on an internal message queue. And finally, send it back to client when response returned from the database.
- Reusable, shared, worker processes: Responsible for picking work off the message queue and execute that work and finally places the response on an output message queue.

Hybrid architecture of Client/Server database suffer from queue latencies, which have an adversely affect on other users. The hybrid architecture of Client/Server database is shown in Fig. 4.6 given below. Some of the examples of such architectures are Oracle7i and Oracle8i/9i.

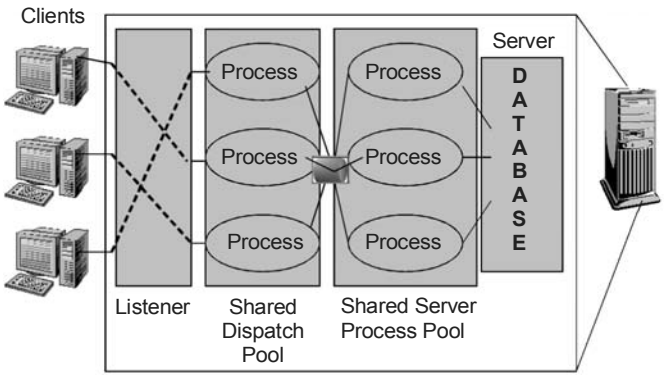

Fig.4.6: Hybrid Architecture

4.4 DATABASE MIDDLEWARE COMPONENT

As we have already discussed in Client/Server architecture that communication middleware software provides the means through which clients and servers communicate to perform specific actions. This middleware software is divided into three main components. As shown in the Fig. 4.7 given below:

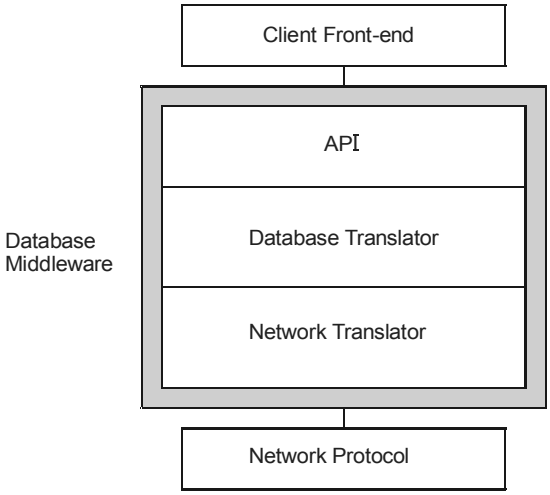

Fig.4.7: Database Middleware Components

- Application programming interface.
- Database translator.
- Network translator.

These components (or their functions) are generally distributed among several software layers that are interchangeable in a plug and play fashion.

The *application-programming interface* is public to the client application. The programmer interacts with the middleware through the APIs provided by middleware software. The middleware API allows the programmer to write generic SQL code instead of code specific to each database server. In other words, the middleware API allows the client process to be database independent. Such independence means that the server can be changed without requiring that the client applications be completely rewritten.

The *database translator* translates the SQL requests into the specific database server syntax. The database translator layer takes the generic SQL request and maps it to the database server's SQL protocol. Because a database server might have some non-standard features, the database translator layer may opt to translate the generic SQL request into the specific format used by the database server.

If the SQL request uses data from two different database servers, the database translator layer will take care of communicating with each server, retrieving the data using the common format expected by the client application.

The network translator manages the network communication protocols. Remember that database server can use any of the network protocols. If a client application taps into the two databases, one that uses TCP/IP and another that uses IPX/SPX, the network layer handles all the communications detail of each database transparently to the client application. Figure 4.8 illustrates the interaction between client and middleware database components.

Existence of these three middleware components reveals some benefits of using middleware software; according to that clients can:

- Access multiple (and quite different) databases
- Be database-server-independent
- Be network-protocol-independent

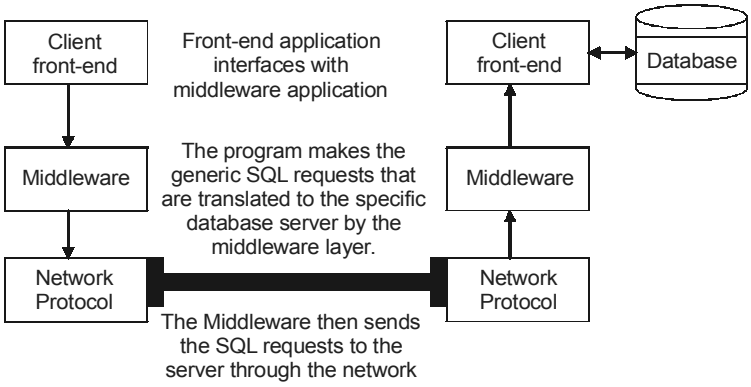

Fig.4.8: Interaction Between Client/Server Middleware Components

4.5 ACCESS TO MULTIPLE DATABASES

To understand how the three components of middleware database work together, lets us see how a client accesses two different databases servers. The Fig. 4.9 shows a client application request data from an oracle database server (Oracle Corporation) and a SQL Server database server (Microsoft Corporation). The Oracle database server uses SQL *Net as its communications protocol with the client; the SQL Server uses Named Pipes as the communications protocol. SQL *Net, a proprietary solution limited to Oracle database, is used by Oracle to send SQL request over a network. Named Pipes is a inter-process communication (IPC) protocol common to multitasking operating systems such as UNIX and OS/2, and it is used in SQL Server to manage both client and server communications across the network.

As per the Fig. 4.9, it is notable that the Oracle server runs under the UNIX operating system and uses TCP/IP as its network protocol. The SQL Server runs under the Windows NT operating system and uses NetBIOS as its network protocol. In this case, the client application uses a generic SQL query to access data in two tables: an Oracle table and a SQL Server table. The database translator layer of middleware software contains two modules, one for each database server type to be accessed.

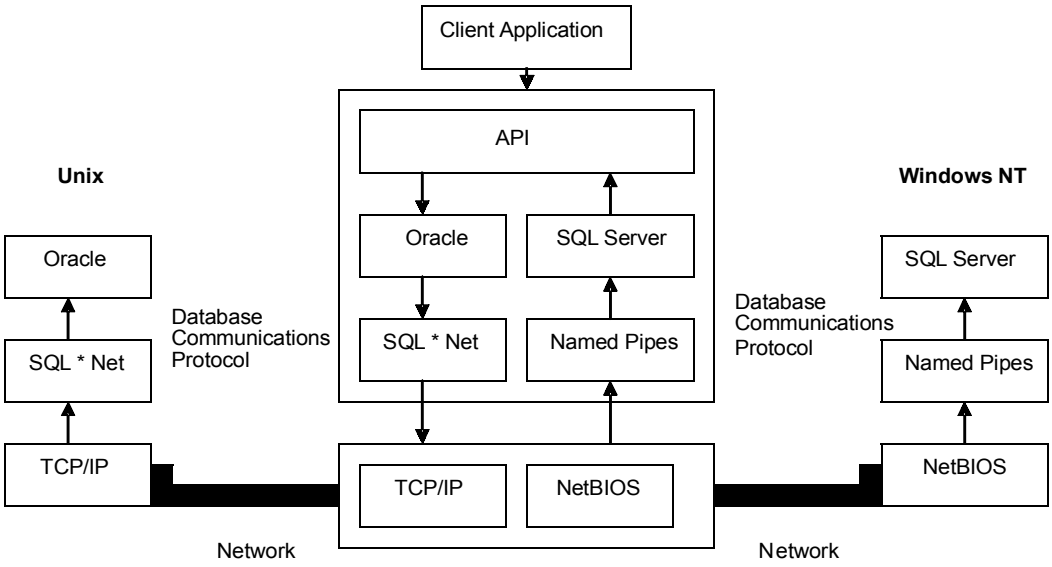

Fig.4.9: Multiple Database Server Access Through Middleware

Each module handles the details of each database communications protocol. The network translator layer takes care of using the correct network protocol to access each database. When the data from the query are returned, they are presented in a format common to the client application. The end user or programmer need not be aware of the details of data retrieval from the servers. Actually, the end user might not even know where the data reside or from what type of DBMS the data were retrieved.

4.6 DISTRIBUTED CLIENT/SERVER DATABASE SYSTEMS

Data Distribution

Distributed data and *distributed processing* are terms used widely in the word of Client/ Server computing. The differences in these two can be easily understood by the two figures 4.10(a) and 4.10(b). Distributed data refers to the basic data stored in the server, which is distributed to different members of the work team. While distributed processing refers to the way different tasks are organized among members of the work team. If a set of information handling tasks is thought of as a single step-by-step process and is split among

the members of the work-team so that they can handle the steps more efficiently, that process is distributed.

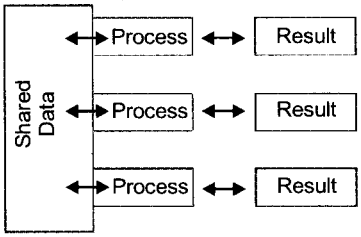

Fig.4.10(*a*): Distributed Data

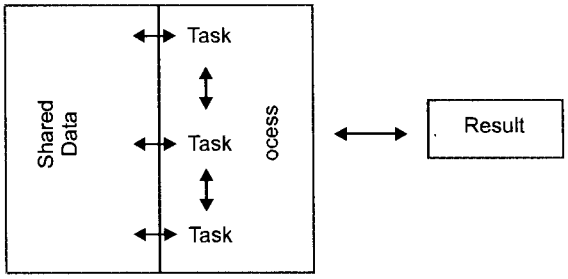

Fig.4.10(*b*): Distributed Processing

Here's an example: Imagine that a customer's ordering and payment information is stored in a central customer record on the server. This record is accessed by many departments in various locations throughout the company (accounting and shipping/ receiving, to name two). Thus the data is an example of distributed data. In addition, because accounting and shipping/receiving work with the data in unique but related ways in order to accomplish a specific goal (updating the customer record), their activities are an example of distributed processing.

Distributed Client/Server database system must have some characteristics that are discussed in this section. The location of data is transparent to the user. The data can be located in the local PC, the department server, or in a mainframe across the country. The data can also be distributed among different locations and among different databases using the same or even the different data models.

The data in database can be partitioned in several ways. And the partitioned data may be allocated (process of data allocation describes where to locate the data, some data allocation strategies are: Centralized, partitioned, replicated) in several different ways, and the data may be replicated in several nodes see the Fig. 4.11.

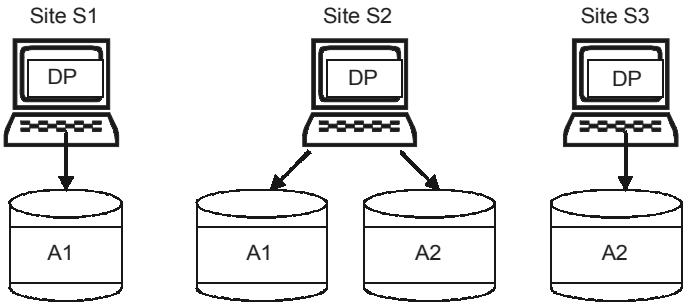

Fig.4.11: Data Replication

For example, suppose database A is divided into two fragments A1 and A2. Within a replicated distributed database, the scenario depicted in figure 4.11 is possible: fragment A1 is stored at sites S1 and S2, while fragment A2 is stored at sites S2 and S3. The network can be a LAN, a MAN, or a WAN. The user does not need to know the data location, how to get there, or what protocols are used to get there.

- Data can be accessed and manipulated by the end user at any time in many ways. Data accessibility increases because end users are able to access data directly and easily, usually by pointing and clicking in their GUI-based system. End user can manipulate data in several ways, depending on their information needs. For example, one user may want to have a report generated in a certain format, whereas another user may prefer to use graphical presentations. Powerful applications stored on the end user's side allow access and manipulation of data in a way that were never before available. The data request is processed on the server side; the data formatting and presentation are done on the client side.
- The processing of data (retrieval, storage, validation, formatting, presentation and so on) is distributed among multiple computers. For example, suppose that a distributed Client/Server system is used to access data from three DBMSs located at different sites. If a user requests a report, the client front-end will issue a SQL request to the DBMS server. The database server will take care of locating the data; retrieving it from the different locations, assembling it, and sending it back to the client. In this scenario, the processing of the data access and retrieval.

4.7 DISTRIBUTED DBMS

Client/Server database is commonly known for having distributed database capabilities. But is not necessarily able to fulfil the entire required Client/Server characteristics that are in need for particular system. Client/Server architecture refers to the way in which computers interact to form a system. The Client/Server architecture features a user of resources, or a client, and a provider of resources, or a server. The Client/Server architecture

can be used to implement a DBMS in which the client is Transaction Processor and the server is the Data Processor. Client/Server interaction in a DDBMS are carefully scripted. The client (TP) interacts with the end use and sends a request to the server (DP). The server receives, schedules, and executes the request, selecting only those records that are needed by the client. The server then sends the data to the client only when the client requests the data. The database management system must be able to manage the distribution of data among multiple nodes. The DBMS must provide distributed database transparency features like:

- Distribution transparency.
- Transaction transparency.
- Failure transparency.
- Performance transparency.
- Heterogeneity transparency.

Number of relational DBMS, which are started as a centralized system with its components like user interface and application programs were moved to the client side. With standard language SQL, creates a logical dividing point between client and server. Hence, the query and transaction functionality remained at the server side. When DBMS access is required, the program establishes a connection to the DBMS; which is on the server side and once the connection is created, the client program can communicate with the DBMS.

Exactly how to divide the DBMS functionality between client and server has not yet been established. Different approaches have been proposed. One possibility is to include functionality of a centralized DBMS at the server level. A number of relational DBMS concepts have taken this approach, where an SQL server is provided to the clients. Each client must then formulate the appropriate SQL queries and provide the user interface and programming language interface functions. Since SQL is a relational standard, various SQL servers possibly provided by different vendors, can accept SQL commands. The client may also refer to a data dictionary that includes information on the distribution of data among the various SQL servers, as well as modules for decomposing a global query into a number of local queries that can be executed at the various sites. Interaction between client and server might proceed as follows during the processing of an SQL query:

- The client passes a user query and decomposes it into a number of independent site queries. Each site query is sent to the appropriate server site.
- Each server process the local query and sends the resulting relation to the client site.
- The client site combines the results of the subqueries to produce the result of the originally submitted query.

In this approach, the SQL server has also been called a transaction server or a Database Processor (DP) or a back-end machine, whereas the client has been called Application

Processor (AP) or a front-end machine. The interaction between client and server can be specified by the user at the client level or via a specialized DBMS client module that is part of DBMS package. For example, the user may know what data is stored in each server, break-down a query request into site subqueries manually, and submit individual subqueries to the various sites. The resulting tables may be combined explicitly by a further user query at the client level. The alternative is to have the client module undertake these actions automatically.

In a typical DBMS, it is customary to divide the software module into three levels:

L1: The server software is responsible for local data management at site, much like centralized DBMS software.

L2: The client software is responsible for most of the distributions; it access data distribution information from the DBMS catalog and process all request that requires access to more than one site. It also handles all user interfaces.

L3: The communication software (sometimes in conjunction with a distributed operating system) provides the communication primitives that are used by the client to transmit commands and data among the various sites as needed. This is not strictly part of the DBMS, but it provides essential communication primitives and services.

The client is responsible for generating a distributed execution plan for a multisite query or transaction and for supervising distributed execution by sending commands to servers. These commands include local queries and transaction to be executed, as well as commands to transmit data to other clients or servers. Hence, client software should be included at any site where multisite queries are submitted. Another function controlled by the client (or coordinator) is that of ensuring consistency of replicated copies of a data item by employing distributed (or global) concurrency control techniques. The client must also ensure the atomicity of global transaction by performing global recovery when certain sites fail. One possible function of the client is to hide the details of data distribution from the user; that is, it enables the user to write global queries and transactions as through the database were centralized, without having to specify the sites at which the data references in the query or transaction resides. This property is called distributed transparency. Some DDBMSs do not provide distribution transparency, instead requiring that users beware of the details of data distribution. In fact, there is some resemblance in between Client/Server and DDBMS. The Client/Server system distributes data processing among several sites, whether as the DDBMS distributes the data at different locations, involving some complimentary and overlapping functions. DDBMS use distributed processing to access data at multiple sites.

4.8 WEB/DATABASE SYSTEM FOR CLIENT/SERVER APPLICATIONS

Nowadays, almost all the MNC's providing information and performing all their activities online through internet or intranet. In this way, the information retrieval becomes quick

and easier. It is obvious that all kinds of information the corporate world is providing through web pages. Also through links on home page they provides the facilities to enter into the corporate intranet, whether it is finance, human resource, sales, manufacturing or the marketing department. Departmental information as well as services can be accessed from web pages Even the web is powerful and flexible tool for supporting corporate requirements but they provides a limited capability for maintaining a large, change base of data. To get effectiveness on Intranet/Internet the organizations are connecting the web services to a database with its own database management systems.

Web-database integration has been illustrated in Fig. 4.12 shown below; a client machine that runs a web browser issues a request for information in the form of a URL (Uniform Resource Locator) reference. This reference triggers a program at the web server that issues the correct database command to a database server. The output returned to the web server is converted into a HTML format and returned to the web browser.

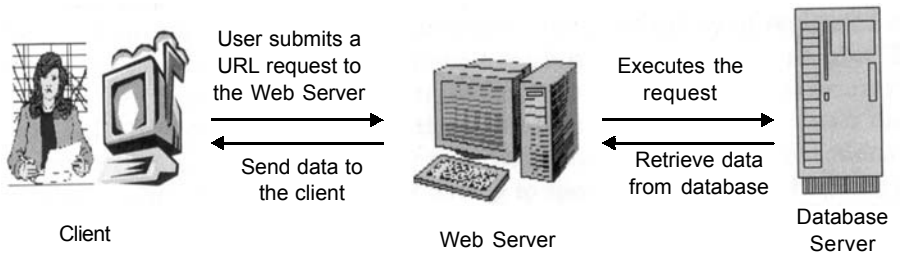

Fig.4.12: Web Database System Integration

4.8.1 Web/Database vs. Traditional Database

The section given below lists the advantages of a web/database system compared to a more traditional database approach.

- *Administration:* The only connection to the database server is the web server. The addition of a new type of database server does not require configuration of all the requisite drivers and interfaces at each type of client machine. Instead, it is only necessary for the web server to be able to convert between HTML and the database interface.
- *Deployment:* Browsers are already available across almost all platforms. Which relieves the developer of the need to implement graphical user interface across multiple customer machines and operating systems? In addition, developers can assume that customers already have and will be able to use browsers as soon as the internet web server is available. Avoiding deployment issues such as installation and synchronized activation.

- *Speed:* Large portion of the normal development cycle, such as development and client design, do not apply to web-based projects. In addition, the text based tags of HTML allow for rapid modification, making it easy to continually improve the look and feel of the application based on the ser feedback. By contrast, changing form or content of a typical graphical-based application can be a substantial task.
- *Information presentation:* Hypermedia base of the web enables the application developers to employ whatever information structure is best for given application, including the use of hierarchical formats in which progressive levels of detail are available to the user.

The section follows lists the disadvantages of a web/database system compared to a more traditional database approach.

- *Functionality*: Compared to the functionality available with a sophisticated graphical ser interface, a typical web browser interface is limited.
- *Operations*: The nature of the HTTP is such that each interaction between a browser and a server is a separate transaction, independent of prior or future exchanges. Typically, the web server keeps no information between transactions to track the states of the user.

EXERCISE 4

1. Explain the DBMS concept in Client/Server architecture in brief.
2. Is the structure of the data important to consider for processing environments? Discuss.
3. If the two servers process the same database, can it be called a Client/Server system? Explain with example.
4. One disadvantage of Client/Server system concerns control in a Database Management environment – explain the disadvantages with an example.
5. "Resource sharing architecture is not suitable for transaction processing in Client/Server environment." Discuss.
6. Compare the object-oriented and relational database management system.
7. Discuss some types of database utilities, tools and their functions.
8. What are the responsibilities of the DBA and the database designers? Also discuss the capabilities that should be provided by a DBMS.